BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-024293
(43)Date of publication of application : 25.01.2002

(51)Int.Cl. G06F 17/30
H04L 12/56

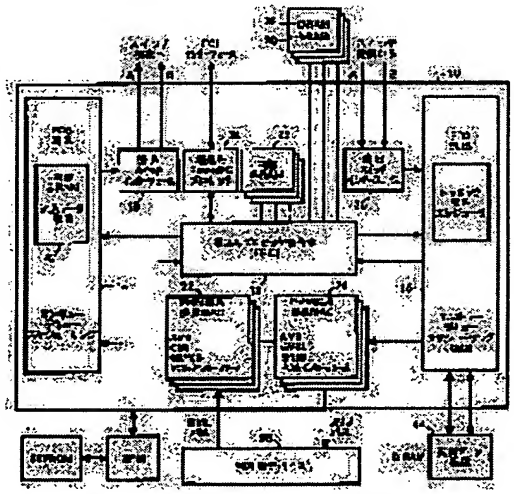
(21)Application number : 2001-104687
(22)Date of filing : 03.04.2001

(71)Applicant : INTERNATL BUSINESS MACH CORP <IBM>
(72)Inventor : CALVIGNAC JEAN LOUIS
HEDDES MARCO C
JEFFERIES CLARK DEBUS
PATEL PIYUSH CHUNILAL
RINALDI MARK ANTHONY

(30)Priority
Priority number : 2000 545100 Priority date : 06.04.2000 Priority country : US

(54) PATTERN RANGE COMPARISON IMPLEMENTING METHOD, COMPUTER-READABLE MEDIUM, AND DEVICE

(57)Abstract:
PROBLEM TO BE SOLVED: To provide a new data structure, a method, and a device for a software management tree(SMT) which provide a control point processor.
SOLUTION: This retrieving mechanism reduces the storage space of a node by using only a forward pointer together with a next bit or bit group to be tested next. Multiple retrieval is unnecessary and filter rules for an application are processed to make it possible to connect various filter rules. Two patterns of the same length are stored in respective leaves to define range comparison. Final comparing operation is comparison within a range or comparison below a mask. Through the comparison within the range, it is decided whether or not an input key is within the range defined by the two patterns. Through the comparison below the mask, various bits in the input key are compared with various bits in a 1st leaf pattern under the mask specified with the 2nd leaf pattern.



LEGAL STATUS

[Date of request for examination] 03.04.2001

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報**（Ａ）　(11)特許出願公開番号
特開2002－24293
（P2002－24293A）
(43)公開日　平成14年１月25日(2002.1.25)

(51)Int.Cl.'	識別記号	ＦＩ		テーマコード'(参考)
Ｇ０６Ｆ　17/30	４１２	Ｇ０６Ｆ　17/30	４１２	５Ｂ０７５
	１１０		１１０Ｃ	５Ｋ０３０
	４１４		４１４Ａ	
Ｈ０４Ｌ　12/56	１００	Ｈ０４Ｌ　12/56	１００Ｚ	

審査請求　有　　請求項の数42　ＯＬ　（全 26 頁）

(21)出願番号　特願2001－104687(P2001－104687)

(22)出願日　平成13年４月３日(2001.4.3)

(31)優先権主張番号　０９／５４５１００

(32)優先日　平成12年４月６日(2000.4.6)

(33)優先権主張国　米国（ＵＳ）

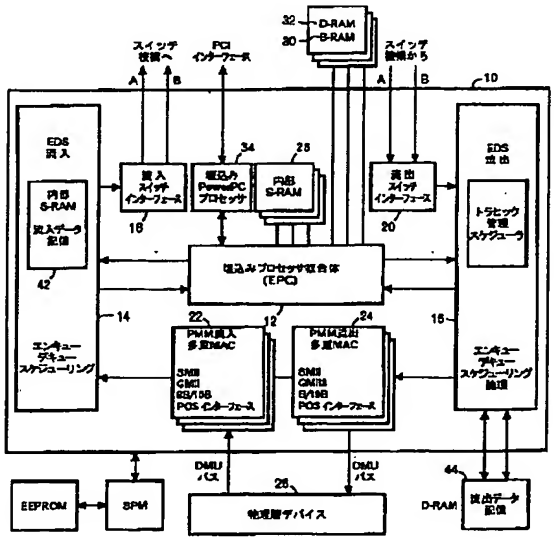
(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレーション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国10504、ニューヨーク州　アーモンク　（番地なし）

(74)代理人　100086243
弁理士　坂口　博　（外２名）

最終頁に続く

(54)【発明の名称】　パターン範囲比較実行方法、コンピュータ可読媒体および装置

(57)【要約】　（修正有）
【課題】　制御ポイント・プロセッサを提供するソフトウェア管理ツリー（ＳＭＴ）のための新奇なデータ構造、方法および装置を提供すること。
【解決手段】　本検索機構は、テストすべき次のビットまたはビット・グループとともに前方ポインタのみを使用し、ノードの記憶スペースを減少させる。また複数の検索は不要で、アプリケーションに対する複数のフィルタ・ルールを処理し、様々なフィルタ・ルールを結びつけることも可能にする。同じ長さの２つのパターンが、それぞれのリーフに記憶されて範囲比較を定義する。最終比較動作は、範囲下の比較かマスク下の比較のいずれかである。範囲下の比較では、入力キーが２つのパターンによって定義された範囲内にあるかが判定される。マスク下の比較では、入力キー内の諸ビットが、第２リーフ・パターンで特定されたマスク下で、第１リーフ・パターン内の諸ビットと比較される。



1

【特許請求の範囲】

【請求項１】ソフトウェア管理ツリー内の可変長検索キーに対して、コンピュータ処理デバイスによりパターン範囲比較を実行する方法であって、

検索文字列として入力キーを読込む行為と、

検索キーの最上位Ｎビットを、検索ツリーの複数のルート・ノードを表すテーブルに対するインデックスとして使用する行為であって、非空のエントリのそれぞれが検索ツリー内の次の分岐またはリーフに対するポインタを含む行為と、

非空のテーブル・エントリ内の前記ポインタが、対応する検索ツリーのリーフまたは次の分岐のどちらを指すかを判定する行為と、

前記ポインタが前記対応する検索ツリーのリーフを指さない場合に、次の分岐内容を読込む行為と、

対応する検索ツリーのリーフに到達したときにリーフ内容を読込み、前記リーフ内の一対のパターンと前記検索キーを比較して、前記一対のリーフ・パターンによって定義された範囲が前記検索キーを含むかどうかを判定する行為と、

前記リーフ・パターンが前記検索キーを含む場合に、見つかったリーフの内容を、要求するアプリケーションに戻す行為とを含む方法。

【請求項２】検索キーを形成するためにプログラム可能ハッシュ関数を使用して入力キーをハッシュすることをさらに含む、請求項１に記載のパターン範囲比較を実行する方法。

【請求項３】検索ツリーの複数のルート・ノードを表す前記テーブルが2^N個のエントリを含む、請求項１に記載のパターン範囲比較を実行する方法。

【請求項４】前記コンピュータ処理デバイスがネットワーク・プロセッサである、請求項１に記載のパターン範囲比較を実行する方法。

【請求項５】前記対応する検索ツリーの前記次の分岐の内容が他の次の分岐を指す、請求項１に記載のパターン範囲比較を実行する方法。

【請求項６】前記次の分岐の内容が前記対応する検索ツリーの前記リーフを指す、請求項１に記載のパターン範囲比較を実行する方法。

【請求項７】前記リーフ・パターンが前記検索キーを含まず、他のリーフに対するポインタを含まない場合に、成功表示を戻さないことをさらに含む、請求項１に記載のパターン範囲比較を実行する方法。

【請求項８】前記テーブルに対する前記インデックスが空エントリを指している場合に、成功表示を戻さないことをさらに含む、請求項１に記載のパターン範囲比較を実行する方法。

【請求項９】前記検索キーにカラー・レジスタの内容を追加して最終検索キーを提供することをさらに含む、請求項１に記載のパターン範囲比較を実行する方法。

2

【請求項１０】前記検索キーに０の文字列を追加して最終検索キーを提供することをさらに含む、請求項１に記載のパターン範囲比較を実行する方法。

【請求項１１】前記一対のパターンによって定義される範囲内にあるかどうかを判定するためにチェックされる整数として前記検索キー内の諸ビットが扱われる、範囲下の比較動作を、一対のパターンを比較する前記行為が含む、請求項１に記載のパターン範囲比較を実行する方法。

【請求項１２】第２リーフ・パターン内に指定されたマスク下で第１リーフ・パターン内の諸ビットと前記検索キー内の諸ビットが比較される、マスク下の比較動作を、一対のパターンを比較する前記行為が含む、請求項１に記載の方法。

【請求項１３】前記リーフが他のリーフに対する連鎖ポインタを含む場合に、他のリーフ内に記憶された一対のパターンを読込み、前記パターンと前記検索キーを比較する行為と、

記憶された前記パターンが前記検索キーを含まず、連鎖内の次のリーフに対するポインタを含まない場合に、成功表示を戻さない行為とをさらに含む、請求項１に記載のパターン範囲比較を実行する方法。

【請求項１４】前記リーフが他のリーフに対する連鎖ポインタを含む場合に、他のリーフに記憶された一対のパターンを読込み、前記パターンと前記検索キーを比較する行為と、

記憶された前記パターンが前記検索キーを含む場合に、成功表示を戻す行為とをさらに含む、請求項１に記載のパターン範囲比較を実行する方法。

【請求項１５】ソフトウェア管理ツリー内の可変長検索キーに対してパターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体であって、

検索されるべきパターンまたはキーと、

検索ツリーに対する第１アドレス・ロケーションを記憶する直接テーブルと、

前記検索ツリー内の分岐をそれぞれ表す複数のパターン検索制御ブロックと、

各エントリに関連付けられた少なくとも１つの範囲比較を指定する比較テーブルと、

複数のリーフであって、前記検索キーと比較する一対のパターンを各リーフが記憶するリーフとを含むコンピュータ可読媒体。

【請求項１６】ツリー検索メモリを管理するルックアップ定義テーブルをさらに含む、請求項１５に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

【請求項１７】前記ルックアップ定義テーブルが、前記ツリーが中にある物理メモリ、前記キーおよびリーフのサイズ、ならびに実行されるべき検索のタイプ、を定義するエントリを含む、請求項１５に記載の、パターン範

囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

【請求項１８】前記ルックアップ定義テーブルが、複数のメモリに実施される、請求項１５に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

【請求項１９】直接テーブル・エントリのフォーマットが、少なくとも１つの検索制御ブロック、次のパターン検索制御ブロックを指す次のパターン・アドレス、リーフまたは結果を指すリーフ制御ブロック・アドレス、テストすべき次のビットまたは諸ビット、および直接リーフを含む、請求項１５に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

【請求項２０】パターン検索制御ブロックのフォーマットが、少なくとも１つの検索制御ブロック、次のパターン検索制御ブロックを指す次のパターン・アドレス、リーフまたは結果を指すリーフ制御ブロック・アドレス、およびテストすべき次のビットまたは諸ビットを含む、請求項１５に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

【請求項２１】前記比較テーブルが、少なくとも１つの範囲比較を定義するエントリを含み、各範囲比較が、前記フィールドの第１ビットの位置であるオフセット・パラメータおよび前記フィールドのビット長である長さパラメータによって定義される、請求項１５に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

【請求項２２】リーフ・データ構造が、少なくとも１つのリーフ連鎖ポインタ、プレフィクス長、前記検索キーと比較すべき一対のパターン、および可変ユーザ・データを含む、請求項１５に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

【請求項２３】前記直接リーフが、直接テーブル・エントリに直接記憶され、検索制御ブロック、および検索キーと比較すべき一対のパターンを含む、請求項１５に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

【請求項２４】パターン検索制御ブロックが、前記検索ツリー内のリーフ・パターンが変化する位置に挿入される、請求項１５に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

【請求項２５】パターン検索制御ブロックが、幅１および高さ１により定義される形を有し、少なくとも３６ビットのライン長を有するメモリ内に記憶される、請求項１５に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

【請求項２６】ソフトウェア管理ツリー内の可変長検索キーに対してパターン範囲比較を実行するための、半導

体基板上に組立てられた装置であって、

複数のプロトコル・プロセッサ、およびフレーム処理を提供する内部制御ポイント・プロセッサを含む組込みプロセッサ複合体と、

各プロトコル・プロセッサにアクセス可能で、高速パターン検索、データ操作、およびフレーム分析を提供する複数のハードウェア・アクセラレータ・コプロセッサと、

少なくとも１つの検索ツリーを表す複数のデータ構造を記憶する複数のプログラム可能メモリ・デバイスであって、前記データ構造が、直接テーブル、パターン検索制御ブロック、比較テーブル、および前記検索キーと比較すべき一対のパターンを含むリーフを含むプログラム可能メモリ・デバイスと、

前記複数のメモリ・デバイスに対する各プロトコル・プロセッサのアクセスを制御する制御メモリ・アービタとを備える装置。

【請求項２７】メモリ読込みおよび書込みならびにメモリ範囲チェックを含むツリー検索命令を実行するプロトコル・プロセッサの実行と並行して動作するツリー検索エンジンをさらに備える、請求項２６に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

【請求項２８】前記複数のメモリ・デバイスが、少なくとも１つの内部静的ランダム・アクセス・メモリ、外部静的ランダム・アクセス・メモリ、および外部動的ランダム・アクセス・メモリをさらに備える、請求項２６に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

【請求項２９】前記制御メモリ・アービタが、前記複数のプロトコル・プロセッサと前記複数のメモリ・デバイスの間でメモリ・サイクルを割り振ることによって制御メモリ動作を管理する、請求項２６に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

【請求項３０】各プロトコル・プロセッサが、プライマリ・データ・バッファ、スクラッチ・パッド・データ・バッファおよびデータ記憶動作のための制御レジスタを備える、請求項２６に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

【請求項３１】前記検索キーに対して幾何学的ハッシュ関数を実行するハッシュ・ボックス・コンポーネントをさらに備える、請求項２６に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

【請求項３２】プログラム可能検索キー・レジスタおよびプログラム可能ハッシュ化キー・レジスタをさらに備える、請求項２６に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

【請求項３３】複数の独立した検索ツリー間で単一のテーブル・データ構造を共用することを可能にするための

5

プログラム可能カラー・キー・レジスタをさらに備える、請求項３２に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

【請求項３４】前記カラー・レジスタの内容が、もしイネーブルであれば、前記ハッシュ出力に追加されて最終ハッシュ化キーを生成する、請求項３３に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

【請求項３５】前記カラー・レジスタがイネーブルでない場合に、最終ハッシュ化キー生成のために前記ハッシュ出力に同数の０を追加する、請求項３３に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

【請求項３６】ソフトウェア管理ツリー内の可変長検索キーに対してパターン範囲比較を実行するコンピュータ・プログラム・プロダクトを含むコンピュータ可読媒体であって、

入力キーを検索文字列として読込むプログラム命令と、

前記検索キーの最上位Ｎビットを、検索ツリーの複数のルート・ノードを表すテーブルに対するインデックスとして使用するプログラム命令であって、それぞれの非空のエントリが、前記検索ツリー内の次の分岐またはリーフに対するポインタを含むプログラム命令と、

非空のテーブル・エントリ内の前記ポインタが、前記対応する検索ツリーのリーフまたは次の分岐のどちらを指すかを判定するプログラム命令と、

前記ポインタが、前記対応する検索ツリーの前記リーフを指さない場合に、前記次の分岐内容を読込むプログラム命令と、

対応する検索ツリーの前記リーフに到達したときに前記リーフ内容を読込み、前記リーフ内の一対のパターンと前記検索キーを比較して前記一対のリーフ・パターンによって定義された範囲が前記検索キーを含むかどうかを判定するプログラム命令と、

前記リーフ・パターンが前記検索キーを含む場合に、見つかった前記リーフの内容を前記要求するアプリケーションに戻すプログラム命令とを含むコンピュータ可読媒体。

【請求項３７】前記テーブルに対する前記インデックスが空のエントリを指している場合に、成功表示を戻さないプログラム命令をさらに含む、請求項３６に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

【請求項３８】カラー・レジスタの内容を前記検索キーに追加して最終検索キーを提供するプログラム命令をさらに含む、請求項３６に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

【請求項３９】前記検索キーに０の文字列を追加して最終検索キーを提供するプログラム命令をさらに含む、請求項３６に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

6

【請求項４０】前記検索キー内の前記諸ビットが、前記一対のパターンによって定義される範囲内にあるかどうかを判定するためにチェックされる整数として扱われる、範囲下の比較動作を実行するプログラム命令をさらに含む、請求項３６に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

【請求項４１】前記検索キー内の諸ビットが、第１リーフ位置の諸ビットと、第２リーフ・パターンに指定されたマスク下で比較される、マスク下の比較動作を実行するプログラム命令をさらに含む、請求項３６に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

【請求項４２】前記リーフが他のリーフに対する連鎖ポインタを含む場合に、他のリーフに記憶された一対のパターンを読込み、前記パターンと前記検索キーを比較するプログラム命令と、

記憶された前記パターンが、前記検索キーを含まず、前記連鎖内の次のリーフに対するポインタを含まない場合に、成功表示を戻さないプログラム命令とをさらに含む、請求項３６に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、一般にパターン・マッチング・アルゴリズムに関し、より詳細には、ネットワーク処理デバイス内に実施することのできるソフトウェア管理ツリー検索アルゴリズムに関する。

【０００２】関連出願の相互参照

本出願は、同時係属で、共通の譲渡人に譲渡された、米国特許出願番号０９／３８４６９１で１９９９年８月２７日出願の「Network Processor Processing Complex and Methods」という表題の特許出願と、同じく出願番号０９／５４３５３の「Full Match (FM) Search Algorithm Implementation for a Network Processor」という表題の特許出願と、同じ出願番号０９／５４４９９２の「LongestPrefix Match (LPM) Algorithm Implementation for a Network Processor」という表題の特許出願とに関係し、必要により参照されたい。

【０００３】

【従来の技術】より一層複雑なタスクをメディア・スピードでサポートする、ハードウェアによって統合された処理に対する要求が、ネットワーク・プロセッサの創造につながった。ネットワーク・プロセッサは、１組の組み込まれた、プログラム可能プロトコル・プロセッサおよび補足的システム・コプロセッサを介して、機能柔軟性を有するワイヤスピード・フレーム処理／転送機能を提供する。ネットワーク・プロセッサは、マイクロプロセッサが今日のパーソナル・コンピュータに対するのと同様に、ネットワークに対する基本的なネットワーク構

7

成要素になると期待されている。ネットワーク・プロセッサは、複数のデータ・ストリームのリアルタイム処理、高度なセキュリティの提供、ならびにＩＰパケット処理／転送機能を提供する。さらにこのネットワーク・プロセッサは、並列分散処理およびパイプライン処理設計などの高度なアーキテクチャを介してスピードの改善を可能にする。これらの機能は、効率的検索エンジン、データ処理スループット向上を可能にし、複雑なタスクの迅速な実行を可能にする。ネットワーク・プロセッサのプログラム可能な特徴は、新たなカスタムＡＳＩＣ（Application Specific Integrated Circuit：特定用途向けＩＣ）設計を必要とすることなく、新たなプロトコルおよび技術を実施するためのより容易な移行方法を、ネットワーク・プロダクト開発者に提供する。

【０００４】ネットワーク・プロセッサは、インターネットまたは企業ネットワークのプロバイダのための相互接続ソリューションを開発するために、高度にカスタマイズ可能でスケーラブルな技術を提供する。ネットワーク・プロセッサは、ローエンドのスタンドアロン・デバイスから大きなマルチラック・ソリューションまで、広範囲のソリューションのための基礎を提供する。この種のスケーリングは、高性能、非ブロック化パケット・ルーティング・スイッチ技術、および他の業界スイッチ技術に適合することのできる、ＩＢＭ CorporationのＤＡＳＬ（Data Aligned Serial Link）インタフェースなどの独占インタフェースの使用により達成することができる。

【０００５】プログラム可能通信集積回路としてネットワーク・プロセッサは、非常に効率的なパケット分類、フレーム毎のマルチテーブル・ルックアップ、パケット修正、待ち行列／ポリシ管理、その他のパケット処理機能を提供する。ネットワーク・プロセッサは、スイッチング・エンジン、検索エンジン、フレーム・プロセッサおよびEthernet ＭＡＣを１つのデバイス上に統合して、大容量メディア・ウェイト・スイッチング・フレームを必要とする顧客のニーズを任意のプロトコル層のフレーム内容に基づいてサポートする。

【０００６】ハードウェア・アクセラレータは、フレーム転送、フレーム・フィルタリングおよびフレーム変更を実行する。複雑な範囲およびアクション仕様を有する数百のルールを執行するネットワーク・プロセッサの能力は、フィルタリング機能に対する新たなベンチマークを設定し、ネットワーク・プロセッサに基づくシステムを大容量サーバ・ファーム・アプリケーションに一意に適合させる。

【０００７】ネットワーク・プロセッサと共に開発される典型的システムは、分散ソフトウェア・モデルを用い、それぞれのプログラム可能ネットワーク・プロセッサは諸タスクを同時に実行する。いくつかの機能は、制御ポイント（ＣＰ）プロセッサ内で実行され、このプロセッサはネットワーク・プロセッサの内部または外部でよい。ＣＰは、層２および層３のルーティング・プロトコル、ならびに層４および層５のネットワーク・アプリケーションおよびシステム管理に対するサポートを提供する。ワイヤスピード転送／フィルタリング機能は、ネットワーク・プロセッサ・ハードウェアと常駐ピココードの組合わせによって実行される。

8

【０００８】いくつかの相互接続されたノードを備える通信ネットワークにおいて、データは１つのノードから任意の他のノードまたはネットワークに送ることができる。ルータと呼ばれる特別なノードは、データをその宛先に転送することを担当する。通信ネットワークを介して送られる任意のデータは、宛先アドレスに関する情報を、一般にヘッダの一部として含む。各ルータは、この情報または少なくともその一部を、内部に記憶されたアドレスのリストと比較する。記憶されたアドレスと宛先アドレスの間に一致が見つかる場合、ルータは宛先ノードに至るパスを確立する。ネットワーク・サイズおよび構造次第で、データはその宛先に直接転送されるか、または他の中間ルータに送られる。国際標準化機構（ＩＳＯ）は、ルータが部分アドレスに対するルーティング情報を記憶するルーティング標準を公表した。次いでルータは、そのルータが自分のデータベース内に有する、最も一致する部分アドレスにパケットを送る。ＩＳＯ標準は、所与の桁数または所与のヘッダ長を用いて、階層構造のノードを構築することを可能にする。メイン・ルータはそのアドレスの最初の部分によってアドレス指定され、サブルータは中間の部分、最終の宛先はアドレスの最後の桁によってアドレス指定される。したがってどのルータも、データが送られる階層レベルに割り当てられた桁を読めば充分である。

【０００９】受け取ったパケットのルーティングは、付随するアドレス文字列に基づく。アドレス文字列はデータベース内の検索キーとして用いられ、このデータベースは、アドレス文字列、ならびにパケット配信の際にどのルータが次であるかなど、他の適切な詳細を含む。データベースは、ルーティング・テーブルと呼ばれるが、現在のルータと次のルータの間のリンクは、パケットの進行中の次のホップと呼ばれる。ルーティング・テーブル検索処理は、アドレスの構造およびテーブルの構成に依存する。例えば、８ビットより小さいサイズで非階層構造を有する検索キーであれば、一連のアドレス・エントリとして構成されたルーティング・テーブル内で最も効率的に見つかる。検索キーは、正しいエントリを見つけるための、テーブル内のインデックスとして用いられる。より大きいサイズ、例えば３２ビットの検索キーに対して、対応するルーティング・テーブルは１０，０００より多いエントリを有する可能性がある。インデックスによって直接検索される単純なテーブルとしてデータベースを構成すれば、テーブルのほとんどが空なので、

9

大量のメモリ・スペースを無駄にする。

【００１０】従来型のルータは、検索処理をいくつかのステップに分解する。第１ステップは、ルータが宛先ホスト・コンピュータに直接接続されているかどうかを決定することである。この場合、メッセージは宛先から１ホップであり、その方向にルーティングされるべきである。宛先コンピュータがルータに直接接続されていない場合、次のステップは宛先ネットワークのトポロジ的方向を決定することである。トポロジ的レイアウトから方向が決定された場合、メッセージはその進路にルーティングされる。そうでない場合、最終ステップはデフォルト・リンクに沿ってメッセージをルーティングすることである。

【００１１】通常第１ステップは、ルータに直接接続されたホスト・コンピュータの３２ビット・アドレスを含むテーブル全体の直線検索を用いて実行される。局所的トポロジを反映して、アドレス・テーブル内の各エントリは、アドレス指定されたコンピュータに直接つながる対応する出力インタフェースに接続される。宛先アドレスがルータに受け取られると、３２ビット全体がテーブル内の宛先アドレスのそれぞれと比較される。一致が見つかる場合、メッセージは、指定されたルータ・インタフェースを介して対応する宛先に直接送られる。

【００１２】第２ステップ、すなわち宛先ネットワークの方向を決定するステップは、ネットワーク・アドレス数のためにそのようなテーブルを管理し使用するのが困難になるので、普通はテーブル全体の直線検索によっては実行されない。従来技術においてアドレス文字列が３レベル階層のネットワーク・アドレス、サブネット・アドレス、およびホスト識別に準拠したとき、ルータは、ハッシング、パトリシア・ツリー検索、およびマルチレベル検索など、いくつかのよく知られている手法の１つを用いて決定を実行した。ハッシングにおいて、ハッシュ関数はアドレスのネットワーク部分を少なくし、小さい管理可能なインデックスを生成する。ハッシュ・インデックスは、ハッシュ・テーブルに索引付けすること、および一致するハッシュ・エントリを検索することのために用いられる。ハッシュ・テーブルの各ハッシュ・エントリには、対応するネットワークのトポロジ的方向を指す出力インタフェースのアドレスが対応している。ハッシュ・ネットワーク部分とハッシュ・エントリの間の一致が見つかる場合、メッセージは対応するインタフェースおよび宛先ネットワークに向けられる。

【００１３】ハッシングは、大きい管理不能なフィールドを小さい管理可能なインデックスに減少させる。しかしこの処理において、２つ以上のフィールドが同じハッシュ・インデックスを生成する可能性がある。これらのフィールドはハッシュ・テーブル内の同じ位置に記憶されなければならないので、この事象は衝突と呼ばれる。衝突中のエントリを区別するためには追加の検索が必要

10

である。したがって衝突は、ハッシング検索を用いることから得られる効率を減少させ、すべての許されるアドレスが単一のインデックスに減少する最悪の場合、ハッシングは検索処理として事実上無益になる。

【００１４】パトリシア・ツリー検索は、ハッシング法が直面した衝突を回避する。この検索方法では、すべてのアドレス文字列、および関連するルート情報などの付随情報がバイナリ・ツリー内に記憶される必要がある。アドレス文字列中の最上位ビット位置から開始して、この検索処理はアドレスとツリー・ノードをビット毎に比較する。一致したビット値により、この検索が左または右のどちらの子ノードに行くかが導かれ、アドレスの次のビットに対して処理が繰り返される。検索時間は、記憶された最長のアドレス文字列のサイズに比例する。パトリシア・ツリー検索では、平均検索時間と最悪の場合の検索時間の間の相違はそれほど大きくない。さらに、ルーティング・テーブルは非常に効率的に構成される。この検索は、ハッシング法の同等のルーティング・テーブルより小さいメモリしか必要としない。パトリシア・ツリー検索は、最悪の場合の検索をハッシング法よりうまく処理するが、ほとんどの場合、一致を見つけるのにかなりより長くかかる。したがって多数の従来型ルータは、ハッシングおよびパトリシア・ツリー検索の組合わせを用いる。この組合わせは、マルチレベル検索と呼ばれる。

【００１５】マルチレベル検索は、ハッシングをパトリシア・ツリー検索と結びつける。キャッシュは、直近に、かつおそらく最も共通にルーティングされるネットワーク・アドレスのサブセットを含むハッシュ・テーブルを記憶するが、パトリシア・ツリーは、ネットワーク・アドレスの完全なセットを記憶する。メッセージが受け取られると、宛先アドレスがテーブル上にハッシングされる。所定の時間以内にその宛先アドレスが見つからない場合、もし記憶されていればアドレスが見つかることを保証するパトリシア・ツリー検索エンジンに、このアドレスが渡される。

【００１６】従来技術には、固定一致ツリー、最長プレフィクス一致ツリー、およびソフトウェア管理ツリーを含む、いくつかの知られているツリー検索アルゴリズムがある。固定一致ツリーは、層２のEthernet ＭＡＣテーブルなど、正確な一致を必要とする固定サイズのパターンに対して用いられる。最長プレフィクス一致ツリーは、ＩＰサブネット転送など、部分的一致しか必要としない可変長パターンに対して用いられる。ソフトウェア管理ツリーは、フィルタ・ルールなど、範囲またはビット・マスクと定義されるパターンに対して用いられる。一般にルックアップは、ツリー検索エンジン（ＴＳＥ）の助けとともに実行される。

【００１７】

【発明が解決しようとする課題】本発明の一目的は、複

11

数の検索を必要とすることなく、アプリケーションに対する複数のフィルタ・ルールを処理するユニークで効率的な機構を提供することである。

【００１８】本発明の他の目的は、後方ポインタに対する記憶を必要とせず、テストすべき次のビットまたはビット・グループとともに前方ポインタだけしか使用しない検索機構を提供し、それによって、ノードに対する記憶スペースを減少させることである。

【００１９】

【課題を解決するための手段】本発明は、制御ポイントによって定義される検索機構に従うツリー構造を作り出す機構を提供するソフトウェア管理ツリー（Software Managed Tree：ＳＭＴ）のための新奇なデータ構造を述べる。この一例示的機構は、ＩＰソース・アドレス（ＩＰＳＡ）、ＩＰ宛先アドレス（ＩＰＤＡ）、ソース・ポート（ＳＰ）、宛先ポート（ＤＰ）、および通信プロトコルを含むインターネット・プロトコル（ＩＰ）５タプル・フィルタリング・テーブルである。完全一致または最長プレフィクス一致ツリーとは対照的に、ＳＭＴツリーは範囲比較に対するサポートを可能にする。例えば、ソース・ポートがｘとｙの範囲内になければならないことを指定するためにリーフを用いることができる。この手法により、効率的な記憶／検索時間を有する非常に効率的で単純な実施が可能になる。この手法により、様々なフィルタ・ルールを結びつけることもまた可能になる。

【００２０】

【発明の実施の形態】本発明が組み込まれるネットワーク・プロセッサの文脈において本発明を述べる。ネットワーク・プロセッサ１０は、単一チップ上のプログラム可能スイッチングおよびルーティング・システムであり、そのアーキテクチャは図１に示されている。このネットワーク・プロセッサは、10/100 Ethernet、Gigabit Ethernet、およびPacket Over SONET（ＰＯＳ）用のメディア・インタフェース、ならびにスイッチ・インタフェースに対する接続用のデータ配列シリアル・リンク（data aligned serial links：ＤＡＳＬ）を提供する。内部ハードウェア・アクセラレータは、性能および効率を向上する。組込みプロセッサ複合体（ＥＰＣ）１２は、プロトコル・プロセッサ、ならびにフレーム処理、構成および管理サポートのための内部制御ポイント・プロセッサを含む。

【００２１】最大N個の並列プロトコル・プロセッサが利用可能である。１６プロトコル・プロセッサの実施形態では、１６，３８４ワードの内部ピココード命令記憶および３２，７６８ワードの外部ピココード命令記憶が利用可能であり、２，１２８ＭＩＰＳ（million instructions per second：百万命令／秒）の合計処理能力を提供する。さらに各プロトコル・プロセッサは、高速パターン検索、データ操作、内部チップ管理機能、フレーム分析、およびデータ・プリフェッチング・サポートを提供するM個のハードウェア・アクセラレータ・コプロセッサに対するアクセスを有する。好ましい一実施形態においてプロトコル・プロセッサのための制御記憶は、内部および外部記憶の両方、すなわち即時アクセスのための３２Ｋ内部ＳＲＡＭ（static random access memory：静的ランダム・アクセス・メモリ）２８、高速アクセスのための外部ＺＢＴ（zero bus turnaround）ＳＲＡＭ３０、および大きい記憶要件のための外部ＤＤＲ（double data rate）ＤＲＡＭ（dynamic random access memory：動的ランダム・アクセス・メモリ）３２によって提供される。

12

【００２２】接続された制御ポイント・プロセッサ３４上で動作する組込みハードウェア・アクセラレータを、前処理アルゴリズムと共に用いると、ネットワーク・プロセッサ１０は、複雑な範囲、優先順位、およびアクション仕様を有する百以上のフィルタ・ルールを介して、ワイヤスピードでフレームを処理することができる。これにより、ネットワーク・プロセッサに基づくシステムが、ゲートウェイ、サーバ・ファーム・アプリケーション、および混合トラヒックの処理に関連付けられたフィルタリング・タスクによく適合するようになる。

【００２３】制御ポイント・ソフトウェアは、ネットワーク管理者がフィルタ・ルールを一貫性のあるユーザフレンドリなインタフェースにいつ入力するかをチェックする自動ロジックを提供する。安定性理論に基づく新規なフロー制御を用いて、ネットワーク・プロセッサ１０は、ＴＣＰ（Transmission Control Protocol：伝送制御プロトコル）崩壊することなく、通常使用されるランダム早期廃棄法（random early discard method）より高い割合の一時的オーバーサブスクリプション（oversubscription）に耐える。ネットワーク・プロセッサ１０は、帯域幅を自動的に割り振ることにより細分化されたサービスもまた配信し、瞬間的または仮定したトラヒック統計に基づいて数十のしきい値を設定することの効果を予測しなければならないことからネットワーク管理者を解放する。

【００２４】単一のネットワーク・プロセッサ１０は、最大４０個のFast Ethernetまたは４個のGigabit Ethernetポートに対するメディア・スピード・スイッチングを提供する。このネットワーク・プロセッサは、ＯＣ－４８ｃ、ＯＣ－４８、４つのＯＣ－１２、または１６個のＯＣ－３ポートをサポートするように構成することもまたできる。スケーラビリティに関しては、２つのネットワーク・プロセッサを相互接続するために２つの３．５ＧｂｐｓシリアルＤＡＳＬリンクを使用して、その結果ポート密度を２倍にするか、またはスイッチ機構を接続して、最大６４個のネットワーク・プロセッサを有するスイッチング・ソリューションを作り出すことができる。１つのプライマリおよび１つのセカンダリという、

13

２つのＤＡＳＬリンクは、システム可用性を向上させるための冗長スイッチ機構に対する接続もまた提供することができる。

【００２５】ネットワーク・プロセッサ１０の一例示的実施形態は、図１に示される次の主要なセクションを含む。

１．最大１６個のプログラム可能プロセッサおよびコプロセッサを含む組込みプロセッサ複合体（ＥＰＣ）．１２。

２．Ethernet物理層デバイスからスイッチ機構に移動するフレームのためのエンキュー・デキュー・スケジューリング論理回路１４（ＥＤＳ流入）。

３．スイッチ機構からEthernet物理層デバイスに移動するフレームのためのエンキュー・デキュー・スケジューリング論理回路１６（ＥＤＳ流出）。

４．他のネットワーク・プロセッサまたは中間スイッチに対する相互接続のための流入スイッチ・インタフェース（スイッチ流入）１８および流出スイッチ・インタフェース（スイッチ流出）２０ＤＡＳＬリンク。

５．EthernetまたはＰＯＳ物理層デバイス２６からフレームを受信する物理ＭＡＣマルチプレクサ２２（ＰＭＭ流入）、およびEthernetまたはＰＯＳ物理層デバイス２６にフレームを送信する物理ＭＡＣマルチプレクサ２４（ＰＭＭ流出）。

【００２６】図２は、組込みプロセッサ複合体のための一例示的実施形態を示す。この組込みプロセッサ複合体は、２１２８ＭＩＰＳの処理能力を提供する１６個のプロトコル・プロセッサを含む。各プロトコル・プロセッサ４０は、３ステージのパイプライン（フェッチ、デコードおよび実行）、汎用レジスタ、専用レジスタ、８命令キャッシュ、専用演算論理ユニット（ＡＬＵ）およびすべて１３３ＭＨｚで動作するコプロセッサを含む。プロトコル・プロセッサの２つ、すなわち誘導フレームを処理するためのプロトコル・プロセッサ（guided frame handler：誘導フレーム・ハンドラ）、および制御メモリ内のルックアップ・データを構築するためのプロトコル・プロセッサ（generic tree handler：総称的ツリー・ハンドラ）は特別である。

【００２７】図３は、プログラム可能プロトコル・プロセッサ４０のそれぞれに関連付けられたコプロセッサは次の機能を提供する。

１．データ記憶コプロセッサ６４は、フレーム・バッファ・メモリ４２、４４（流入／流出方向）のインタフェースとなり、直接メモリ・アクセス（ＤＭＡ）機能を提供する。

２．チェックサム・コプロセッサ６２は、ヘッダ・チェックサムを計算する。

３．エンキュー・コプロセッサ６６は、２５６ビット・ワーク・レジスタに対するアクセスを制御し、キー・フレーム・パラメータを含む。このコプロセッサは、完了ユニット４６とのインタフェースとなり、フレームをスイッチおよびターゲット・ポート・キューにエンキューする。

14

４．インタフェース・コプロセッサは、デバッグまたは統計収集のために、内部レジスタ、カウンタ、およびメモリに対するすべてのプロトコル・プロセッサ・アクセスを提供する。

５．文字列コピー・コプロセッサは、ＥＰＣ中のデータの効率的移動を可能にする。

６．カウンタ・コプロセッサは、プロトコル・プロセッサ４０のためのカウンタ更新を管理する。

７．ポリシ・コプロセッサは、フロー制御情報を調査し、予め割り振られた帯域幅との一致をチェックする。

【００２８】ハードウェア・アクセラレータ４８は、フレーム転送、フレーム・フィルタリング、フレーム変更およびツリー検索を実行する。ネットワーク・プロセッサに組込まれた他の機能には、革新的なフィルタ・ルール処理、ハッシュ関数およびフロー制御が含まれる。

【００２９】プロトコル・プロセッサ４０は、複雑な範囲およびアクション仕様を有する百以上のフレーム・フィルタ・ルールを執行することができる。フィルタリングは、ネットワーク・セキュリティにとって本質的であり、ネットワーク・プロセッサ・ハードウェア支援４８は、これらの複雑なルール・セットのワイヤスピードでの執行を提供する。フィルタ・ルールはフレームを拒否または許可するか、あるいはインターネット・プロトコル（ＩＰ）ヘッダ情報に基づいてサービスの品質（ＱｏＳ）を割り振ることができる。前処理ルールのための制御ポイント・ソフトウェアは、ロジック・エラーを自動的に訂正する。論理的に正しいルール・セットが入力された後に、パケット・ヘッダ情報からキーが形成され、ネットワーク・プロセッサのソフトウェア管理ツリーを用いてワイヤスピードでテストされる。

【００３０】理想的なランダム・ハッシュを上回るために、幾何学的ハッシュ関数はＩＰヘッダ内の統計的構造を利用する。その結果、衝突率の低下により、他の分解検索なしで、完全一致テーブル内の高速ルックアップが可能になる。

【００３１】プロトコル・プロセッサの実行と並行して動作しながら、ツリー検索エンジン７０は、（メモリ読取り、書込みまたは読み書きを含む）ツリー検索命令、メモリ範囲チェックおよび不正メモリ・アクセス通知を実行する。図１４は、ツリー検索エンジンの一例示的実施形態を示す。

【００３２】ネットワーク・プロセッサ１０の中では２つのシステム制御オプションが利用可能である。内部プロセッサ３４は、システムのための制御ポイント（ＣＰ）プロセッサとして機能することができ、あるいは外部プロセッサは、初期化および構成のための４つのEthe

15

metマクロの１つに接続することができる。ＣＰプロセッサ３４は、誘導フレームと呼ばれる特別なEthernetフレームを構築することによって、諸ネットワーク・プロセッサ中の他のプロセッサ・エンティティと通信する。誘導フレームは、ＤＡＳＬリンクを越えて他のデバイスに転送することができ、単一のEthernetポートに接続された１つのＣＰプロセッサが、そのサブシステム内に含まれたすべてのネットワーク・プロセッサ・デバイスと通信し制御することを可能にする。各ネットワーク・プロセッサの内部プロセッサ３４もまた、別々の３２ビットＰＣＩバスを用いて通信することができる。

【００３３】ネットワーク・プロセッサ１０は普通、サブシステム・ボード上に常駐し、プロトコル層（すなわち、層２、層３、層４およびより高い層）フレーム処理を提供する。ＣＰサブシステム内のＣＰプロセッサ３４上で動作するソフトウェアは、管理およびルート発見機能を提供する。ＣＰコード、プロトコル・プロセッサ上で動作するピココード、および誘導フレーム・ハンドラ上で動作するピココードは、このシステムの初期化、転送パスの維持、およびシステムの管理を提供する。分散システムとして、ＣＰおよび各ネットワーク・プロセッサ・サブシステムは、効率および性能の向上のために誘導フレームを用いて並列動作し通信する複数のプロセッサを含む。

【００３４】データ・フレームは、ＰＭＭ２２によってメディアから受信され、データ記憶バッファ４２に転送される。ＰＭＭは、受信処理中にＣＲＣチェックおよびフレーム妥当性検査もまた実行する。ディスパッチャ５０は、最大６４バイトのフレーム情報をフレーム・ルックアップに利用可能なプロトコル・プロセッサ４０に送る。クラシファイヤ・ハードウェア支援４８は、制御データを供給してフレーム・フォーマットを識別する。プロトコル・プロセッサ４０は、固定一致ツリー、最長プレフィクス一致ツリー、ソフトウェア管理ツリーを含む、適用すべきツリー検索アルゴリズムを決定するために制御データを使用する。

【００３５】ルックアップは、ツリー検索エンジン（ＴＳＥ）７０の支援で実行される。ＴＳＥ７０は、制御メモリ７２のアクセスを実行し、プロトコル・プロセッサが実行を続けることを可能にする。制御メモリ７２は、すべてのテーブル、カウンタおよびピココードに必要とされる他の任意のデータを記憶する。効率のために、制御メモリ・アービタ（arbiter）５２は、プロトコル・プロセッサ４０と様々なオンチップおよびオフチップ制御メモリ・オプション５４の間にメモリ・サイクルを割り振ることによって制御メモリ動作を管理する。

【００３６】プロトコル・プロセッサ４０は、データ記憶動作のためのプライマリ・データ・バッファ、スクラッチ・パッド・データ・バッファおよび諸制御レジスタ（集合的に７２）を含む。一致が見つかると、ＶＬＡＮ

16

ヘッダ挿入あるいはオーバーレイなどの流入フレーム変更を適用することができる。これらの変更は、ＥＰＣ１２によっては実行されない。代わりに、ハードウェア・フラグが設定されている場合、流入スイッチ・インタフェース・ハードウェア１８が変更を実行する。他のフレーム変更は、流入データ記憶４２の中に保持されるフレーム内容を修正することによってピココードおよびデータ記憶コプロセッサ６４により達成することができる。

【００３７】制御データは、スイッチ機構に対してフレームを送る前にスイッチ・ヘッダおよびフレーム・ヘッダを構築するために収集され使用される。制御データは、宛先ポートのフレーム・ルックアップ、マルチキャストまたはユニキャスト動作、および流出フレーム変更、の促進を助けるために、フレームの宛先などのスイッチ情報、ならびに流出ネットワーク・プロセッサのための情報を含む。

【００３８】図４は、一例示的流入／流出フレーム・フローを示す。完了時にエンキュー・コプロセッサ６６が、キュー制御ブロック（ＱＣＢ）７４にフレームをエンキューするために必要なフォーマットを構築し、そのフォーマットを完了ユニット４６に転送する。完了ユニット４６は、最大１６個のプロトコル・プロセッサからスイッチ機構キュー７６までのフレーム順序を保証する。スイッチ機構キューからのフレームは、スイッチ機構７６により送信されるときに、スイッチ・ヘッダおよびフレーム・ヘッダ・バイトが挿入されて、６４バイトのセルに分割される。

【００３９】スイッチ機構７６から受信されたフレームは、再組立て制御ブロック（ＲＣＢ）８０およびＥＤＳ流出４４により提供された情報を用いて流出データ記憶バッファ７８内に置かれ、ＥＰＣ１２にエンキューされる。フレームの一部は、フレーム・ルックアップを実行するために、ディスパッチャ５０によって任意の遊んでいるプロトコル・プロセッサ４０に送られる。フレーム・データは、クラシファイヤ・ハードウェア支援４８からのデータとともにプロトコル・プロセッサ４０に発送される。クラシファイヤ・ハードウェア支援４８は、流入ネットワーク・プロセッサによって作り出されたフレーム制御データを用いて、流出処理のための開始命令アドレスを決定するのを助ける。

【００４０】流出ツリー検索は、流入検索に対してサポートされたのと同じアルゴリズムをサポートする。ＴＳＥ７０でルックアップが実行され、実行を続けるためにプロトコル・プロセッサ４０を解放する。すべての制御メモリ動作は、プロセッサ複合体間のメモリ・アクセスを割り振る制御メモリ・アービタ５２によって管理される。

【００４１】流出フレーム・データは、データ記憶コプロセッサ６４を介してアクセスされる。正常なルックアップの結果は、転送情報、および場合によっては、フレ

ーム変更情報を含む。流出フレーム変更には、ＶＬＡＮヘッダ削除、存続時間（timeto live）増分（ＩＰＸ）または減分（ＩＰ）、ＩＰヘッダ・チェックサム再計算、EthernetフレームＣＲＣオーバーレイ、およびＭＡＣ宛先アドレスまたはソース・アドレスのオーバーレイまたは挿入を含むことができる。ＩＰヘッダ・チェックサムは、チェックサム・コプロセッサ６２によって準備される。変更は、組込みプロセッサ複合体１２によって実行されるのではなく、むしろハードウェア・フラグが作り出されて、ＰＭＭ流出ハードウェア２４が変更を実行する。完了時に、ＥＤＳ流出キュー４４内のフレームをエンキューするために必要なフォーマットを構築するためにエンキュー・コプロセッサ４６が用いられ、そのフォーマットを完了ユニット４６に転送する。完了ユニット４６は、最大１６個のプロトコル・プロセッサから、流出Ethernet ＭＡＣを供給するＥＤＳ流出キュー４４のフレーム順序を保証する。完了したフレームは、最後にＰＭＭ流出ハードウェア２４によって、Ethernet ＭＡＣまたはＰＯＳインタフェース、および物理ポートの外に送られる。

【００４２】図１４に示されたツリー検索エンジン（ＴＳＥ）７０は、情報を記憶し取り出すためにツリー概念を使用する。取り出し、すなわちツリー検索ならびに挿入および削除は、例えばＭＡＣソース・アドレス、ＩＰソース・アドレスとＩＰ宛先アドレスの連結などのビット・パターンであるキーに基づいて行われる。本発明において使用する一例示的ツリー・データ構造１００が図５に示されている。情報は、少なくともキー１０２を含むリーフ１１６、１１８、１２０、１２２と呼ばれる制御ブロック内に記憶される（記憶されるビット・パターンは、実際にはハッシュ化キー１０６である）。リーフは、エージング情報、ユーザ情報などの追加情報もまた含むことができ、このユーザ情報は、ターゲット・ブレード、ターゲット・ポート番号などの転送情報とすることができる。リーフのフォーマットは、ピココードによって定義され、内部または外部制御記憶にオブジェクトが置かれる。

【００４３】ツリーに対する検索アルゴリズムは、キー１０２を含む入力パラメータに対して動作し、そのキーに対してハッシュ１０４を実行し、直接テーブル（ＤＴ）１０８にアクセスし、パターン検索制御ブロック（ＰＳＣＢ）１１０、１１２、１１４を介してツリーをウォークし、リーフ１１６、１１８、１２０、１２２で終了する。各タイプのツリーは、自分自身の検索アルゴリズムを有し、それにより異なるルールにしたがってツリーウォークが行われるようになる。例えば固定一致（ＦＭ）ツリーに対しては、データ構造はパトリシア・ツリーである。リーフが見つかったとき、このリーフは入力キー１０２に一致することができる唯一の可能な候補である。「最終比較」動作が、入力キー１０２とリーフ内に記憶されたパターンを比較する。これが、リーフが入力キー１０２に実際に一致するかどうかを検証する。この検索の結果は、リーフが見つかり一致したときには、成功（ＯＫ）であり、他のすべての場合は失敗（ＫＯ）である。

【００４４】検索動作のための入力は、次のパラメータを含む。

キー（key）

検索または挿入／削除の前に特別ピココード命令を用いて１７６ビット・キーが構築されなければならない。キー・レジスタは１つしかない。しかしツリー検索が開始された後には、ＴＳＥ７０が検索を実行しながら同時に、キー・レジスタを、次の検索のキーを構築するためにピココードによって使用することができる。これは、ＴＳＥ７０がキーをハッシュし、その結果を内部ハッシュ化キー・レジスタ１０６に記憶するからである。

キー長（key length）

この８ビット・レジスタは、キー－１ビットを含む。このレジスタは、キーの構築中にハードウェアによって自動的に更新される。

LUDefIndex

これは、検索が行われるツリーの完全な定義を含むルックアップ定義テーブル（LUDefTable）への８ビット・インデックスである。LUDefTableの内部構造は、図１１に示されている。

ＴＳＲＮｒ

検索結果は、１ビットのツリー検索結果領域ＴＳＲ０またはＴＳＲ１のいずれかに記憶することができる。これはＴＳＲＮｒによって指定される。ＴＳＥが検索している間に、ピココードは他のＴＳＲにアクセスして前の検索結果を分析することができる。

カラー（color）

カラーがイネーブルになったツリー（LUDefTable内で指定される）に対して、１６ビットのカラー・レジスタ１２４の内容が、ハッシュ動作中にキーの中に挿入される。

【００４５】ルックアップ定義テーブルは、ツリー検索メモリを管理するメイン構造である。LUDefTableは、内部メモリ構造であり、ツリーを作り出すための１２８エントリを含む。LUDefTableは、ツリーが中に存在する物理メモリ（例えば、ＤＲＡＭ、ＳＲＡＭ、内部ＲＡＭ）、キャッシュがイネーブルであるかどうか、キーおよびリーフのサイズ、ならびに実行すべき検索アクションのタイプ、を定義するエントリを含む。LUDefTableは、３つの別々のランダム・アクセス・メモリ、すなわち汎用プロセッサ・ツリー・ハンドラ（ＧＴＨ）によってしかアクセス可能でない１つのＲＡＭ、および互いの複製であり、すべてのピコプロセッサによってアクセス可能な２つのＲＡＭとして実施される。

【００４６】ハッシュ関数１０４の出力はいつも、元の

19

入力キー１０２とハッシュ関数１０４の出力の間に１対１対応があるという特性を有する１７６ビットの数である。以下に説明するように、この特性は直接テーブル１０８の後に開始するツリーの深さを最小化する。

【００４７】ツリーのためのカラーがイネーブルである、図５の例の場合、１６ビットのカラー・レジスタ１２４が１７６ビットのハッシュ関数出力に挿入され、フィールドの結果は、HashedKey１０６と呼ばれる１９２ビットの数である。挿入は、直接テーブル１０８のすぐ後ろに行われる。直接テーブル１０８が2^Nのエントリを含む場合、図５に示すように、ビット位置Nに１６ビットのカラー値が挿入される。ハッシュ関数の出力は、挿入されたカラー値と共に、HashedKeyレジスタ１０６内に記憶される。ツリーに対するカラーがディセーブルである場合、１７６ビットのハッシュ関数は修正しないで取られ、１６個の０がハッシュ出力に追加されて、１９２ビットの最終HashedKeyを生成する。

【００４８】カラーは、複数の独立したツリー間で単一の直接テーブル１０８を共用するために使用することができる。例えばカラーの一使用例は、ＭＡＣソース・アドレス（ＳＡ）テーブル内の仮想ローカル・エリア・ネットワーク（ＶＬＡＮ）である。この場合、入力キー１０２はＭＡＣ　ＳＡであり、カラー１２４はＶＬＡＮＩＤである（ＶＬＡＮ　ＩＤは１２ビットなので、カラーの４ビットは使用されない、すなわち０に設定される）。ハッシュ関数１０４の後に使用されているパターンは、４８＋１６＝６４ビットである。カラーは、もはやパターンの一部であり、異なるＶＬＡＮのＭＡＣアドレスを区別する。

【００４９】ハッシュ関数１０４は、自分の出力内のほとんどのエントロピが最高位の諸ビットにあるように定義される。HashedKeyレジスタ１０６の最高位からのＮビットが、直接テーブル（ＤＴ）１０８へのインデックスを計算するために使用される。

【００５０】ＳＭＴツリーに対して、入力キーおよびカラーは合わせて１９２ビットの入力パターンを形成する。この視点では、カラー・レジスタはキー・レジスタに対する拡張と見るべきである。ＳＭＴツリーは、「０」に設定したカラー・イネーブル・ビットを有しなければならない。さらにＳＭＴツリーは、１６ビットのカラー・レジスタ１２４と共に１７６ビットの入力キー１０２を取り、１９２ビットのHashedKey１０６を生成するハッシュ関数を使用しなければならず、それによって、キーが最も左の１７６ビットを形成し、カラーが最も右の１６ビットを形成する。したがってＳＭＴツリーのためのハッシュ関数は、実際にはハッシュ関数ではなく、カラーをキーの拡張として使用する手段である。

【００５１】記憶および検索効率を達成するために、この実施例は次のデータ構造を利用する。

ａ．検索する必要のあるパターン／キー

20

ｂ．直接テーブル（ＤＴ）エントリ

ｃ．パターン検索制御ブロック（ＰＳＣＢ）

ｄ．リーフ

ｅ．最終比較動作

ｆ．比較テーブル・エントリ

【００５２】ＤＴエントリは、キーの最初の「Ｎ」ビットに基づく最初のアドレス・ロケーションである。このエントリは、以下に述べる３つの部分を含む。ＰＳＣＢエントリは中間ノード位置である。リーフ・エントリは、検索結果のためのアドレス・ロケーションである。比較テーブル・エントリは、リーフ／パターン比較パラメータを記述する。ＤＴエントリは、以下にさらに述べるように、幅１かつ高さ１、または幅１かつ高さ２のいずれかにより定義される形を有する。

【００５３】ＰＳＣＢは、ツリー内の分岐である。この好ましい実施形態には、０分岐および１分岐がある。ＰＳＣＢから出る分岐の数は、分岐を指定するために使用されるビットの数に依存する変数である。ｎビットが使用される場合には、ＰＳＣＢで2^nの分岐が定義される。各ＰＳＣＢは、ビット位置ｐにもまたに関連付けられる。ＰＳＣＢから０分岐を介して到達することができるすべてのリーフは、パターン中の位置ｐに「０」を有し、１分岐を介して到達することができるリーフは、位置ｐに「１」を有する。さらにＰＳＣＢから到達することができるすべてのリーフは、ビット０．．．ｐ－１が同一のパターンを常に有する、すなわちこのパターンは、位置ｐにおいて始めて変化する。ＰＳＣＢに関連付けられたビット位置は、前のＰＳＣＢまたはＤＴエントリ内に記憶され、ＮＢＴ（すなわちnext bit to test：テストすべき次のビット）と呼ばれる。

【００５４】したがってＰＳＣＢは、ツリー内のリーフ・パターンが変化する位置にしか挿入されない。ＰＳＣＢの数、したがって検索性能は、ツリー内のリーフ数にのみ依存し、パターンの長さには依存しないので、これにより効率的検索動作が可能になる。ＰＳＣＢレジスタ・フォーマットは、図１２に示されている。

【００５５】ＳＭＴツリーにおいてＰＳＣＢの後の第１リーフは、LUDefTable内に定義された形を有しなければならない。リーフ連鎖内の他の任意のリーフは、連鎖ポインタ内の５ビット、すなわちリーフ内のＮＬＡＳＭＴフィールド、により定義される形を有する。

【００５６】ＤＴおよびＰＳＣＢエントリのフォーマットは、同一であり、次の部分を含む。

ａ．ＳＣＢ（search control block：検索制御ブロック）２ビット

ｂ．ＮＰＡ（next pattern address：次のパターン・アドレス）：次のＰＳＣＢアドレスを指す、またはＬＣＢＡ（leaf control block address：リーフ制御ブロック・アドレス）：リーフ／結果を指す。

ｃ．ＮＢＴ（next bit or bits to test：テストすべき

21

次のビット）：テストすべき次の１対または１組の「ｘ」ビット（ｘ＝１またはｎ）とすることができる。テストすべきビット数は、記憶効率に基づき決定される。

ｄ．直接リーフ。

【００５７】この実施例の各エントリは、幅３６ビットであり、現在定義されている３つの可能なエントリ・フォーマットの１つを含む。

ａ．空のＤＴエントリ：ＳＣＢ＝００かつＮＰＡ＝０かつＮＢＴが不正または０。

ｂ．ＮＰＡ／ＮＢＴが妥当：ＳＣＢ＝００かつＮＰＡおよびＮＢＴが妥当。ＤＴエントリに対して、ＮＰＡは第１中間ノードを指し、ＮＢＴはテストすべき１ビットまたは複数ビットを指す。ＰＳＣＢエントリの場合、ＮＰＡは、このトレイル内の他のノードを指す。

ｃ．ＬＣＢＡが妥当：ＳＣＢ＝０１。ＬＣＢＡは、関連付けられたリーフ・アドレス、すなわち検索結果を指す。

ｄ．直接リーフ：ＳＣＢ＝１０かつデータの残りは検索結果またはリーフを含む。リーフ・データの一部は、大きな検索結果記憶をサポートするために、リーフ・アドレスの連鎖を含むことができる。

【００５８】ＤＴかつＰＳＣＢエントリのメモリ割振りに関して：常に２＊＊テストすべきビット数個のアドレスからなる、すなわち対／組であるということを除き、ＳＭＴ　ＰＳＣＢはＳＭＴ　ＤＴエントリと同じ構造を有する。これらのアドレス対または組は、メモリ内に連続的に割り振られ、検索ツリーをウォークするための分岐／ジャンプ・ポインタとして使用される。

【００５９】ＳＭＴツリー内のリーフのフォーマットは、２つのパターンを含む制御情報を含む。この２つのパターンは、範囲比較を定義するために使用される。ＳＭＴ内のリーフは、ＮＬＡＳＭＴフィールドを用いて結びつけることができる。図１３は、ＳＭＴツリーのためのフォーマットを示す。ＰＳＣＢの後の第１リーフに到達するとき、最終比較動作が実行される。これがＯＫを戻すとき、検索は停止する。しかし最終比較がＫＯを戻し、非０のＮＬＡＳＭＴフィールドがあるとき、次のリーフが読込まれ、もう１度最終比較動作が実行される。この処理は、最終比較がＯＫを戻すか、またはＮＬＡＳＭＴフィールドが０に等しくなるまで続き、この場合検索はＫＯで戻る。

【００６０】ソフトウェア管理ツリー検索のための高レベル・アルゴリズム・フローは、次の通りである。

１．ＤＴエントリ読込み

ａ．ＳＣＢ＝００かつＮＰＡおよびＮＢＴが妥当である場合、ＮＰＡおよびＮＢＴを読込み、次のＰＳＣＢアドレスを生成する。

ｂ．ＮＢＴが妥当でなく、直接リーフが妥当である場合、リーフ内容を読込みリーフ評価ステップに行く。

22

ｃ．ＮＢＴが妥当でなく、かつ／またはリーフが存在しない場合、ＫＯ、すなわち検索結果が失敗かつ完了フラグが実行済み、を戻す。

２．リーフ評価：パターン（キー）とリーフ内に記憶されたパターンを比較する。ＳＭＴは、常に２つのパターンを含む。これらの２つのパターンは、範囲比較を定義するために使用される。また、ＳＭＴ内のリーフは、結びつけることもできる（すなわち次のリーフ・アドレスを表すＮＬＡＳＭＴを用いる）。ＰＳＣＢの後の第１リーフがヒットすると、最終比較動作が実行される。これがＯＫ（成功）を戻すとき、検索はＯＫで停止する。しかしこの最終比較がＫＯ（失敗）を戻し、非０の、すなわち妥当なＮＬＡＳＭＴフィールドがあるとき、次のリーフが読込まれ最終比較動作がもう１度実行される。この動作は、最終比較がＯＫ（成功）で戻るか、またはＮＬＡＳＭＴフィールドがもう妥当でなくなるまで続き、この場合、検索はＫＯ（失敗）で戻る。

【００６１】本明細書で述べるビット／レジスタ幅の値は、例示的であり、利用可能なメモリ、性能要件等を最適化するために異なる値に変更することができる。

【００６２】検索は、直接テーブル１０８に対するアクセスで開始する、すなわち直接テーブルからＤＴエントリが読込まれる。ＤＴエントリを読込むために使用されるアドレスは、レジスタ１０６内のHashedKeyの上位Ｎビットから、LUDefTable内で定義されたツリー特性に対して計算される。ＤＴエントリは、ツリーのルートと見ることができる。実際のツリー・データ構造は、ツリー・タイプに依存する。ＳＭＴツリーに対するパトリシア・ツリー・データ構造への拡張が使用される。

【００６３】８エントリＤＴ１０８を使用する例が図６に示されている。検索時間すなわちアクセスされなければならないＰＳＣＢの数は、ＤＴを用いることによって少なくすることができる。したがってＤＴサイズを大きくすることにより、メモリ使用量と検索性能の間のトレードオフが起こり得る。

【００６４】ＤＴエントリを読込んだ結果、そのエントリが１リーフに対するポインタを含むだけで、その後そのリーフ自体を読込まなければならないことになると、性能の理由で非効率である。この状況は、ＤＴエントリにつき単一のリーフ・エントリを多数有するＦＭツリーに対して、非常にしばしば発生する。直接リーフのコンセプトは、より多くのメモリ使用とよりよい性能とのトレードオフを可能にする。

【００６５】ツリーは、直接リーフをイネーブルにすることができ、これはルックアップ定義テーブル（LUDefTable）において指定される。直接リーフをイネーブルにしたツリーとディセーブルにしたツリーの間の相違は、図７に示されている。直接リーフがイネーブルで、ＤＴエントリが単一のリーフを含むとき、このリーフは、ＤＴエントリ自体に直接記憶される。そうでない場合、こ

のＤＴエントリは、リーフに対するポインタを含む。

【００６６】整形は、ＴＳＭの一機能であり、リーフ、ＰＳＣＢのようなオブジェクトがＴＳＭ内にどのように記憶されるかを指定するために用いられる。形は、幅および高さのパラメータによって定義される。オブジェクトの高さは、オブジェクトが記憶される連続的アドレス・ロケーションの数を示す。オブジェクトの幅は、オブジェクトが記憶される連続的バンクの数を示す。幅および高さに対して、ハードウェアは適切な数のロケーションを自動的に読込む。ピココードの視点からは、オブジェクトはアクセスの原子単位である。ＳＲＡＭ内に記憶されるオブジェクトに対する幅は、常に１でなければならない。ＤＲＡＭ内のオブジェクトに対する幅は、１より大きくてよい。単一メモリ・ロケーション中に適合するほど充分小さいオブジェクトは、高さ１、幅１を有すると定義される。直接リーフがディセーブルにされたＤＴエントリの形は、常に（Ｗ＝１、Ｈ＝１）である。ＤＴエントリがＤＲＡＭ（動的ランダム・アクセス・メモリ）内に記憶されるとき、このエントリは正確に６４ビットを占める。直接リーフがイネーブルにされたＤＴエントリの形は、LUDefTable内に指定されたリーフの形に等しい。一般にこれより、ＤＴ１０８によってより多くのメモリが使用されることになる。これはまた、リーフの形がＤＴエントリ・アドレス計算に対して影響を与えることにもなる。

【００６７】ＤＴエントリが読込まれた後、このＤＴエントリが直接リーフを含まず、また空でもないと仮定すると、このＤＴエントリで始まるツリーをウォークすることにより検索が続く。ツリーウォークは、リーフに到達するまで、いくつかのＰＳＣＢ（パターン検索制御ブロック）を通過することができる。

【００６８】ＳＭＴツリー内の検索中にＰＳＣＢに出会うと、ツリー検索エンジン・ハードウェアは、HashedKeyのビットｐの値次第で、０分岐または１分岐に対してツリーウォークを続ける。

【００６９】図１０は、本発明のソフトウェア管理ツリー検索アルゴリズムの処理ロジックを示す。アルゴリズムは、ロジック・ブロック１１００において入力キーの読込みで始まる。ＳＭＴ検索では、入力キーをハッシュ化キーにハッシュするオプションがある。ハッシュ関数は、入力キー内のビット順序を反転することができる。例えばビット１とビット７を交換することができる。ハッシュ関数が使用されるとき、このハッシュ関数は、プログラム可能、すなわちどのビットが交換されるかについてプログラムすることができる。ロジック・ブロック１００２により示されるように、直接テーブルが次に読込まれる。ハッシュ化キーの上位Ｎビット（ここでＮは構成可能である）が直接テーブルへのインデックスとして使用される。読込まれたエントリが空であるとき、終了ブロック１００４により示されるように、検索はＫＯ

（発見なし）を戻す。このエントリが判定ブロック１００６内のリーフを指す場合、処理は、ロジック・ブロック１０１０においてリーフの内容を読込むことで続く。そうでない場合、エントリはＰＳＣＢを指し、ロジック・ブロック１００８に示されるように、ＰＳＣＢの適切な部分が読込まれる。ＳＭＴ検索に関して、ＰＳＣＢはＦＭ　ＰＳＣＢのフォーマットを有することができる。あるいは以下に述べるように、より高度なフォーマットを有することができる。

【００７０】ロジック・ブロック１００８から、処理は判定ブロック１００６に戻り、リーフが見つかったかどうかを判定する。判定ブロック１００６においてリーフが見つかるとき、ロジック１０１０によって示されるように、このリーフが読込まれ、次いでロジック・ブロック１０１２によって示されるように、入力キーと比較される。判定ブロック１０１４においてリーフ内に記憶されたパターンと入力キーの間の一致がある場合、検索はＯＫ（成功）を戻し、終了ブロック１０１８によって示されるように、リーフの内容をアプリケーションに渡す。判定ブロック１０１４において一致がない場合、判定ブロック１０１６において示されるように、次のリーフが存在するかどうかの判定で処理が続く。リーフ連鎖内に次のリーフがある場合、処理はロジック・ブロック１０１０に戻ってそのリーフの内容を読込む。そうではなくて、次のリーフがない場合、終了ブロック１０２０によって示されるように、検索はＫＯ（失敗）を戻す。

【００７１】リーフ・パターンとハッシュ化キーの間の比較動作は、ＦＭとＬＰＭ検索よりも複雑である。リーフは、２つのパターンｐ１およびｐ２を含む。ｐ１およびｐ２内のハッシュ化キーは、それぞれＮ個のフィールドに論理的に分割される。この比較動作は、Ｎ個の部分比較を含み、全体の比較がＯＫを戻すためには、すべてがＯＫ（成功）を戻さなければならない。それぞれの部分比較は、（ｉ）ｐ１が最小、ｐ２が最大を表す（ｉｉ）ｐ１がパターンを表し、ｐ２がマスクを表すという２つのモードにしたがって実行することができる。第１の場合、部分比較はｐ１≦ハッシュ化キー≦ｐ２のときＯＫを戻す。第２の場合、部分比較は、（ハッシュ化キー　ＡＮＤ　ｐ２）＝（ｐ１　ＡＮＤ　ｐ２）のときＯＫを戻す。

【００７２】フィールドの定義（各フィールドのサイズおよび位置）および各フィールドに対する比較モードは、リーフ内にコード化された形態で記憶するか、特別なルックアップ・テーブル内に記憶することができる。この好ましい実施形態では、CompDefTableが使用され、ここでテーブルのインデックスはリーフ内に記憶される。

【００７３】拡張として、ＰＳＣＢは2^b個のエントリから構成することができ、その結果ＰＳＣＢからどのエントリを読込むかをハッシュ化キーのｂビットが選択す

る。これは、より多くのメモリ使用を犠牲にして性能を向上させる。さらにＰＳＣＢは、リーフ比較で述べたのと同じ方法で動作する１つまたは２つのパターン（ｐ１およびｐ２）を各エントリが含むように拡張することができる。例えばＰＳＣＢが、これもまたＰＳＣＢ内に記憶された長さＬのｐ１およびｐ２を含むと仮定する。その場合、ハッシュ化キーから、ＮＢＴによって与えられる位置のＬビットが取られる。これらのＬビットは整数Ｉとして解釈される。ｐ１≦Ｉ≦ｐ２のとき、次のＰＳＣＢのエントリ１が使用され、そうでない場合、次のＰＳＣＢのエントリ０が使用される。

【００７４】ツリー内の検索性能をよくするためにキャッシュを使用することができる。キャッシュの使用は、LUDefTable内でツリー毎にイネーブルにすることができる。検索中に、ツリー検索エンジン７０は、キャッシュ内を最初にチェックし、HashedKeyと一致するリーフが存在するかどうかを判定する。そのようなリーフが見つかる場合、そのリーフが戻され、これ以上の検索は必要でない。そのようなリーフが見つからない場合、通常の検索が始まる。

【００７５】ツリー検索エンジン・ハードウェア７０に対して、キャッシュ・ルックアップは、通常の検索と正確に同一である。したがって入力キーがHashedKeyにハッシュ化され、直接テーブル１０８アクセスが実行される。直接テーブル１０８はキャッシュとしてふるまう。キャッシュ検索がＯＫ（成功）を戻すとき、検索は終了する。そうでない場合、ツリー検索エンジンは、ハッシュ動作が実行されないということを除き、ツリー全体の第２検索を開始する。HashedKeyレジスタ１０６の内容は、再利用される。

【００７６】キャッシュ検索を使用するかどうかは、LUDefTable内で指定することができる。キャッシュ検索がLUDefTableエントリＩを使用し、検索がＫＯ（失敗）で終了する場合、LUDefTableエントリＩ＋１を用いた追加の検索が自動的に始まる。原則的には、これにより複数の検索を結びつけることが可能になるが、LUDefTableエントリＩ＋１の下でツリー全体を記憶することが推奨される。

【００７７】ツリー検索エンジン７０は、ＦＭツリー、ＬＰＭツリーおよびＳＭＴツリーにおいてハードウェア検索動作を提供する。すべてのツリー・タイプに対して、ツリーを初期化し維持するために様々な量のソフトウェアが必要とされる。ＦＭツリーおよびＬＰＭツリーだけが、制御ポイント・プロセッサ３４の介在なしに、リーフの挿入および削除を行うことができる。この機能を用いることにより、スケーラブルな構成が可能になり、さらに、必要であれば、ＣＰ３４がリーフを挿入または削除することが可能になる柔軟性ができる。

【００７８】ＳＭＴは、ＣＰによって定義された検索機構に従うツリー構造をＣＰが作り出す機構を提供する。この一例は、ＩＰソース・アドレス（ＩＰＳＡ）、ＩＰ宛先アドレス（ＩＰＤＡ）、ソース・ポート、宛先ポートおよびプロトコルを含むＩＰ５タプル・フィルタリング・テーブルである。ＦＭおよびＬＰＭツリーとは対照的に、ＳＭＴは、範囲に対するサポートを提供する。例えば、ソース・ポートが範囲１００．．．１１０以内になければならないことを指定するために１リーフを用いることができる。

【００７９】ＳＭＴとＦＭ／ＬＰＭツリーの間には次の相違が存在する。

・ＳＭＴは、リーフ内に２つのパターンを常に含む。この２つのパターンは、以下に説明する範囲比較を定義するために使用される。

・ＳＭＴ内のリーフは、ＮＬＡＳＭＴ（次のリーフ・アドレス）フィールドを用いて結びつけることができる。ＰＳＣＢ後の第１リーフがヒットするとき、最終比較動作が実行される。これがＯＫ（成功）を戻すとき、検索はＯＫで停止する。しかしこれがＫＯ（失敗）を戻し、非０のＮＬＡＳＭＴフィールドがあるとき、次のリーフが読込まれ、最終比較動作がもう１度実行される。この処理は、最終比較がＯＫ（成功）を戻すか、またはＮＬＡＳＭＴフィールドが０に等しくなるかのいずれかまで続き、この場合、検索は、ＫＯ（失敗）を戻す。

【００８０】ＦＭおよびＬＰＭツリーとは対照的に、ＳＭＴは各リーフ内に同じ長さの２つのパターンを含む。比較動作の目的で、入力キー（および同様に、リーフ内に記憶された２つのパターン）は、複数のフィールドに論理的に分割することができる。一例が図８に示されている。各フィールドに対して、２つの比較のうち１つを実行することができる。

１．マスク下の比較。入力キー中のビットが、リーフ・パターン１に指定されたマスクの下で、リーフ・パターン０の中のビットと比較される。マスク中の「１」は、入力キー中の対応するビットが、パターン０の中の対応するビットに等しくなければならないことを表し、マスク中の「０」は、入力キー中の対応するビットが、比較に何も影響を与えないことを表す。すべてのビット比較がＯＫの場合に限り、フィールド全体の比較がＯＫになる。

２．範囲下の比較。入力キー中のビットは、整数として扱われ、この整数がチェックされて、最小と最大（両者とも範囲に含まれる）によって与えられる範囲内にあるかどうかが判定される。この場合、フィールド比較はＯＫであり、そうでない場合、フィールド比較はＫＯである。すべてのフィールド比較がＯＫの場合に限り、最終比較全体がＯＫを戻し、そうでない場合、最終比較は、ＫＯを戻す。

【００８１】論理フィールドがどのように定義されるかの定義は、比較定義テーブル（CompDefTable）内に指定され、そのエントリ・フォーマットの一例が図９に与え

られている。省略された場合、CompDefTable内に反対を指定するエントリがない限り、フィールドはマスク下の比較フィールドである。

【００８２】CompDefTable内の各エントリは、１つまたは２つの範囲比較を指定する。以下に説明するように、２つより多い範囲比較を指定するために、複数のエントリを使用することができる。各範囲比較は、２つのパラメータによって定義される。

・フィールドの第１ビットの位置であるオフセット。オフセットは、１６ビット境界になければならず、次の０、１６、３２、４８、６４、８０、９６、１１２、１２８の値を取ることができる。

・ビット内のフィールド長。長さは、次の８、１６、２４、３２の値を取ることができる。

【００８３】例えば、図８に示されたキーに対して、ソース・ポート・フィールドに対する範囲下の比較であれば、６４に設定されたOffset0および１６に設定されたMin/MaxLength0を有し、宛先ポート・フィールドに対する範囲下の比較であれば、８０に設定されたOffset1および１６に設定されたMin/MaxLength1を有する。２つより多い範囲比較が必要である場合、継続ビットが１に設定されなければならず、これにより、CompDefTable内の次のエントリは、追加の１つまたは２つの範囲定義下の比較のために使用されるようになる。ＳＭＴ比較のために使用されるCompDefTable内のインデックスは、リーフ内に指定される。

【００８４】性能の理由から、範囲下の比較を可能な限り使用しないことが推奨される。範囲下の比較はそれぞれ、実行するのに１クロック・サイクル（７．５ナノ秒）余分にかかる。したがって範囲が２の累乗（すなわち１２８～２５５）の場合、範囲下の比較は必要でなく、この種の範囲はマスク下の比較動作を用いて処理することができる。

【００８５】ＳＭＴ最終比較が失敗し、リーフ内のＮＬＡＳＭＴフィールドが非０であるとき、ＴＳＥ７０は、比較がＯＫを戻すか、ＮＬＡＳＭＴが０に等しくなるまで、次のリーフを読込み、最終比較動作をもう１度実行する。

【００８６】本発明は、ハードウェア、ソフトウェア、または両者の組合わせで実現することができる。本明細書に述べた方法を実行するように構成された任意の種類のコンピュータ・システムまたは他の装置が適合する。ハードウェアとソフトウェアの一典型的組合わせは、ロードされ実行されたときに、本明細書に述べた方法を実行するようにコンピュータ・システムを制御する汎用コンピュータ・システムである。本発明は、本明細書に述べた方法の実施を可能にするすべての機能を備え、コンピュータ・システムにロードされるときにこれらの方法を実行することができるコンピュータ・プログラム・プロダクトに組み込むこともまたできる。

【００８７】本文脈内のコンピュータ・プログラム命令またはコンピュータ・プログラムは、情報処理機能を有するシステムに、直接、または次のａ）他の言語、コードまたは表記への変換、ｂ）異なる材質形態での再生産、のいずれか一方または両方が行われた後に特定の機能を実行させることを意図する１組の命令の、任意の言語、コード（すなわちピココード命令）または表記での任意の表現を意味する。

【００８８】本発明の精神および範囲を逸脱することなく、本発明の好ましい実施形態に対して多くの修正が可能であることを、当業者は理解するであろう。さらに本発明のいくつかの機能を、他の機能を対応して使用することなく使用することが可能である。したがって、本発明の範囲は付随の請求の範囲によってのみ定義されるので、好ましい実施形態の前記記述は、本発明の原理を示す目的で提供され、その制限内にはない。

【００８９】まとめとして、本発明の構成に関して以下の事項を開示する。

【００９０】（１）ソフトウェア管理ツリー内の可変長検索キーに対して、コンピュータ処理デバイスによりパターン範囲比較を実行する方法であって、検索文字列として入力キーを読込む行為と、検索キーの最上位Ｎビットを、検索ツリーの複数のルート・ノードを表すテーブルに対するインデックスとして使用する行為であって、非空のエントリのそれぞれが検索ツリー内の次の分岐またはリーフに対するポインタを含む行為と、非空のテーブル・エントリ内の前記ポインタが、対応する検索ツリーのリーフまたは次の分岐のどちらを指すかを判定する行為と、前記ポインタが前記対応する検索ツリーのリーフを指さない場合に、次の分岐内容を読込む行為と、対応する検索ツリーのリーフに到達したときにリーフ内容を読込み、前記リーフ内の一対のパターンと前記検索キーを比較して、前記一対のリーフ・パターンによって定義された範囲が前記検索キーを含むかどうかを判定する行為と、前記リーフ・パターンが前記検索キーを含む場合に、見つかったリーフの内容を、要求するアプリケーションに戻す行為とを含む方法。

（２）検索キーを形成するためにプログラム可能ハッシュ関数を使用して入力キーをハッシュすることをさらに含む、上記（１）に記載のパターン範囲比較を実行する方法。

（３）検索ツリーの複数のルート・ノードを表す前記テーブルが2^N個のエントリを含む、上記（１）に記載のパターン範囲比較を実行する方法。

（４）前記コンピュータ処理デバイスがネットワーク・プロセッサである、上記（１）に記載のパターン範囲比較を実行する方法。

（５）前記対応する検索ツリーの前記次の分岐の内容が他の次の分岐を指す、上記（１）に記載のパターン範囲比較を実行する方法。

（６）前記次の分岐の内容が前記対応する検索ツリーの前記リーフを指す、上記（１）に記載のパターン範囲比較を実行する方法。

（７）前記リーフ・パターンが前記検索キーを含まず、他のリーフに対するポインタを含まない場合に、成功表示を戻さないことをさらに含む、上記（１）に記載のパターン範囲比較を実行する方法。

（８）前記テーブルに対する前記インデックスが空エントリを指している場合に、成功表示を戻さないことをさらに含む、上記（１）に記載のパターン範囲比較を実行する方法。

（９）前記検索キーにカラー・レジスタの内容を追加して最終検索キーを提供することをさらに含む、上記（１）に記載のパターン範囲比較を実行する方法。

（１０）前記検索キーに０の文字列を追加して最終検索キーを提供することをさらに含む、上記（１）に記載のパターン範囲比較を実行する方法。

（１１）前記一対のパターンによって定義される範囲内にあるかどうかを判定するためにチェックされる整数として前記検索キー内の諸ビットが扱われる、範囲下の比較動作を、一対のパターンを比較する前記行為が含む、上記（１）に記載のパターン範囲比較を実行する方法。

（１２）第２リーフ・パターン内に指定されたマスク下で第１リーフ・パターン内の諸ビットと前記検索キー内の諸ビットが比較される、マスク下の比較動作を、一対のパターンを比較する前記行為が含む、上記（１）に記載の方法。

（１３）前記リーフが他のリーフに対する連鎖ポインタを含む場合に、他のリーフ内に記憶された一対のパターンを読込み、前記パターンと前記検索キーを比較する行為と、記憶された前記パターンが前記検索キーを含まず、連鎖内の次のリーフに対するポインタを含まない場合に、成功表示を戻さない行為とをさらに含む、上記（１）に記載のパターン範囲比較を実行する方法。

（１４）前記リーフが他のリーフに対する連鎖ポインタを含む場合に、他のリーフに記憶された一対のパターンを読込み、前記パターンと前記検索キーを比較する行為と、記憶された前記パターンが前記検索キーを含む場合に、成功表示を戻す行為とをさらに含む、上記（１）に記載のパターン範囲比較を実行する方法。

（１５）ソフトウェア管理ツリー内の可変長検索キーに対してパターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体であって、検索されるべきパターンまたはキーと、検索ツリーに対する第１アドレス・ロケーションを記憶する直接テーブルと、前記検索ツリー内の分岐をそれぞれ表す複数のパターン検索制御ブロックと、各エントリに関連付けられた少なくとも１つの範囲比較を指定する比較テーブルと、複数のリーフであって、前記検索キーと比較する一対のパターンを各リーフが記憶するリーフとを含むコンピュータ可読媒体。

（１６）ツリー検索メモリを管理するルックアップ定義テーブルをさらに含む、上記（１５）に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

（１７）前記ルックアップ定義テーブルが、前記ツリーが中にある物理メモリ、前記キーおよびリーフのサイズ、ならびに実行されるべき検索のタイプ、を定義するエントリを含む、上記（１５）に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

（１８）前記ルックアップ定義テーブルが、複数のメモリに実施される、上記（１５）に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

（１９）直接テーブル・エントリのフォーマットが、少なくとも１つの検索制御ブロック、次のパターン検索制御ブロックを指す次のパターン・アドレス、リーフまたは結果を指すリーフ制御ブロック・アドレス、テストすべき次のビットまたは諸ビット、および直接リーフを含む、上記（１５）に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

（２０）パターン検索制御ブロックのフォーマットが、少なくとも１つの検索制御ブロック、次のパターン検索制御ブロックを指す次のパターン・アドレス、リーフまたは結果を指すリーフ制御ブロック・アドレス、およびテストすべき次のビットまたは諸ビットを含む、上記（１５）に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

（２１）前記比較テーブルが、少なくとも１つの範囲比較を定義するエントリを含み、各範囲比較が、前記フィールドの第１ビットの位置であるオフセット・パラメータおよび前記フィールドのビット長である長さパラメータによって定義される、上記（１５）に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

（２２）リーフ・データ構造が、少なくとも１つのリーフ連鎖ポインタ、プレフィクス長、前記検索キーと比較すべき一対のパターン、および可変ユーザ・データを含む、上記（１５）に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

（２３）前記直接リーフが、直接テーブル・エントリに直接記憶され、検索制御ブロック、および検索キーと比較すべき一対のパターンを含む、上記（１５）に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

（２４）パターン検索制御ブロックが、前記検索ツリー内のリーフ・パターンが変化する位置に挿入される、上

31

記（１５）に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

（２５）パターン検索制御ブロックが、幅１および高さ１により定義される形を有し、少なくとも３６ビットのライン長を有するメモリ内に記憶される、上記（１５）に記載の、パターン範囲比較を実行するための複数のデータ構造を含むコンピュータ可読媒体。

（２６）ソフトウェア管理ツリー内の可変長検索キーに対してパターン範囲比較を実行するための、半導体基板上に組立てられた装置であって、複数のプロトコル・プロセッサ、およびフレーム処理を提供する内部制御ポイント・プロセッサを含む組込みプロセッサ複合体と、各プロトコル・プロセッサにアクセス可能で、高速パターン検索、データ操作、およびフレーム分析を提供する複数のハードウェア・アクセラレータ・コプロセッサと、少なくとも１つの検索ツリーを表す複数のデータ構造を記憶する複数のプログラム可能メモリ・デバイスであって、前記データ構造が、直接テーブル、パターン検索制御ブロック、比較テーブル、および前記検索キーと比較すべき一対のパターンを含むリーフを含むプログラム可能メモリ・デバイスと、前記複数のメモリ・デバイスに対する各プロトコル・プロセッサのアクセスを制御する制御メモリ・アービタとを備える装置。

（２７）メモリ読込みおよび書込みならびにメモリ範囲チェックを含むツリー検索命令を実行するプロトコル・プロセッサの実行と並行して動作するツリー検索エンジンをさらに備える、上記（２６）に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

（２８）前記複数のメモリ・デバイスが、少なくとも１つの内部静的ランダム・アクセス・メモリ、外部静的ランダム・アクセス・メモリ、および外部動的ランダム・アクセス・メモリをさらに備える、上記（２６）に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

（２９）前記制御メモリ・アービタが、前記複数のプロトコル・プロセッサと前記複数のメモリ・デバイスの間でメモリ・サイクルを割り振ることによって制御メモリ動作を管理する、上記（２６）に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

（３０）各プロトコル・プロセッサが、プライマリ・データ・バッファ、スクラッチ・パッド・データ・バッファおよびデータ記憶動作のための制御レジスタを備える、上記（２６）に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

（３１）前記検索キーに対して幾何学的ハッシュ関数を実行するハッシュ・ボックス・コンポーネントをさらに備える、上記（２６）に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

32

（３２）プログラム可能検索キー・レジスタおよびプログラム可能ハッシュ化キー・レジスタをさらに備える、上記（２６）に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

（３３）複数の独立した検索ツリー間で単一のテーブル・データ構造を共用することを可能にするためのプログラム可能カラー・キー・レジスタをさらに備える、上記（３２）に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

（３４）前記カラー・レジスタの内容が、もしイネーブルであれば、前記ハッシュ出力に追加されて最終ハッシュ化キーを生成する、上記（３３）に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

（３５）前記カラー・レジスタがイネーブルでない場合に、最終ハッシュ化キー生成のために前記ハッシュ出力に同数の０を追加する、上記（３３）に記載の、パターン範囲比較を実行するための半導体基板上に組立てられた装置。

（３６）ソフトウェア管理ツリー内の可変長検索キーに対してパターン範囲比較を実行するコンピュータ・プログラム・プロダクトを含むコンピュータ可読媒体であって、入力キーを検索文字列として読込むプログラム命令と、前記検索キーの最上位Ｎビットを、検索ツリーの複数のルート・ノードを表すテーブルに対するインデックスとして使用するプログラム命令であって、それぞれの非空のエントリが、前記検索ツリー内の次の分岐またはリーフに対するポインタを含むプログラム命令と、非空のテーブル・エントリ内の前記ポインタが、前記対応する検索ツリーのリーフまたは次の分岐のどちらを指すかを判定するプログラム命令と、前記ポインタが、前記対応する検索ツリーの前記リーフを指さない場合に、前記次の分岐内容を読込むプログラム命令と、対応する検索ツリーの前記リーフに到達したときに前記リーフ内容を読込み、前記リーフ内の一対のパターンと前記検索キーを比較して前記一対のリーフ・パターンによって定義された範囲が前記検索キーを含むかどうかを判定するプログラム命令と、前記リーフ・パターンが前記検索キーを含む場合に、見つかった前記リーフの内容を前記要求するアプリケーションに戻すプログラム命令とを含むコンピュータ可読媒体。

（３７）プログラム可能ハッシュ関数を用いて前記入力キーをハッシュして検索キーを形成するプログラム命令をさらに含む、上記（３６）に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

（３８）検索ツリーの複数のルート・ノードを表す前記テーブルが、2^Nのエントリを含む、上記（３６）に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

（３９）前記コンピュータ処理デバイスがネットワーク

33

・プロセッサである、上記（３６）に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

（４０）前記対応する検索ツリーの前記次の分岐の内容が、他の次の分岐を指す、上記（３６）に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

（４１）前記次の分岐の内容が、前記対応する検索ツリーの前記リーフを指す、上記（３６）に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

（４２）前記リーフ・パターンが、前記検索キーを含まず、他のリーフに対するポインタを含まない場合に、成功表示を戻さないプログラム命令をさらに含む、上記（３６）に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

（４３）前記テーブルに対する前記インデックスが空のエントリを指している場合に、成功表示を戻さないプログラム命令をさらに含む、上記（３６）に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

（４４）カラー・レジスタの内容を前記検索キーに追加して最終検索キーを提供するプログラム命令をさらに含む、上記（３６）に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

（４５）前記検索キーに０の文字列を追加して最終検索キーを提供するプログラム命令をさらに含む、上記（３６）に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

（４６）前記検索キー内の前記諸ビットが、前記一対のパターンによって定義される範囲内にあるかどうかを判定するためにチェックされる整数として扱われる、範囲下の比較動作を実行するプログラム命令をさらに含む、上記（３６）に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

（４７）前記検索キー内の諸ビットが、第１リーフ位置の諸ビットと、第２リーフ・パターンに指定されたマスク下で比較される、マスク下の比較動作を実行するプログラム命令をさらに含む、上記（３６）に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

（４８）前記リーフが他のリーフに対する連鎖ポインタを含む場合に、他のリーフに記憶された一対のパターンを読込み、前記パターンと前記検索キーを比較するプログラム命令と、記憶された前記パターンが、前記検索キーを含まず、前記連鎖内の次のリーフに対するポインタを含まない場合に、成功表示を戻さないプログラム命令とをさらに含む、上記（３６）に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

34

（４９）前記リーフが他のリーフに対する連鎖ポインタを含む場合に、他のリーフに記憶された一対のパターンを読込み、前記パターンと前記検索キーを比較するプログラム命令と、記憶された前記パターンが前記検索キーを含む場合に、成功表示を戻すプログラム命令とをさらに含む、上記（３６）に記載の、パターン範囲比較を実行するコンピュータ・プログラム・プロダクト。

【図面の簡単な説明】

【図１】本発明の好ましい実施形態による、ネットワーク・プロセッサのための一例示的アーキテクチャを示す図である。

【図２】本発明の好ましい実施形態による、組込みプロセッサ複合体のための一例示的実施形態を示す図である。

【図３】本発明の好ましい実施形態による、一例示的プロトコル・プロセッサ構造を示す図である。

【図４】本発明の好ましい実施形態による、一例示的流入／流出フレームの流れを示す図である。

【図５】本発明の好ましい実施形態による、完全一致検索アルゴリズムのためのツリー・データ構造を示す図である。

【図６】本発明の好ましい実施形態による、一例示的データ構造に対する直接テーブルを使用することの効果を示す図である。

【図７】本発明の好ましい実施形態による、一例示的データ構造に対する、直接リーフをイネーブルにすることの効果を示す図である。

【図８】本発明の好ましい実施形態による、ソフトウェア管理ツリー（ＳＭＴ）のための入力キーおよびリーフ・パターン中のフィールドの例を示す図である。

【図９】本発明の好ましい実施形態による、比較定義テーブル・エントリのための一例示的フォーマットを示す図である。

【図１０】本発明の好ましい実施形態による、ソフトウェア管理ツリー検索アルゴリズムの処理ロジックを示す図である。

【図１１】本発明の好ましい実施形態による、一例示的ルックアップ定義テーブルの内部構造を示す図である。

【図１２】ＰＳＣＢレジスタの内部フォーマットを示す図である。

【図１３】ＳＭＴツリーのための固定リーフ・フォーマットを示す図である。

【図１４】本発明の好ましい実施形態による、ツリー検索エンジンのための一例示的アーキテクチャを示す図である。

【符号の説明】

１０　ネットワーク・プロセッサ

１２　組込みプロセッサ複合体

１４　エンキュー・デキュー・スケジューリング論理回路

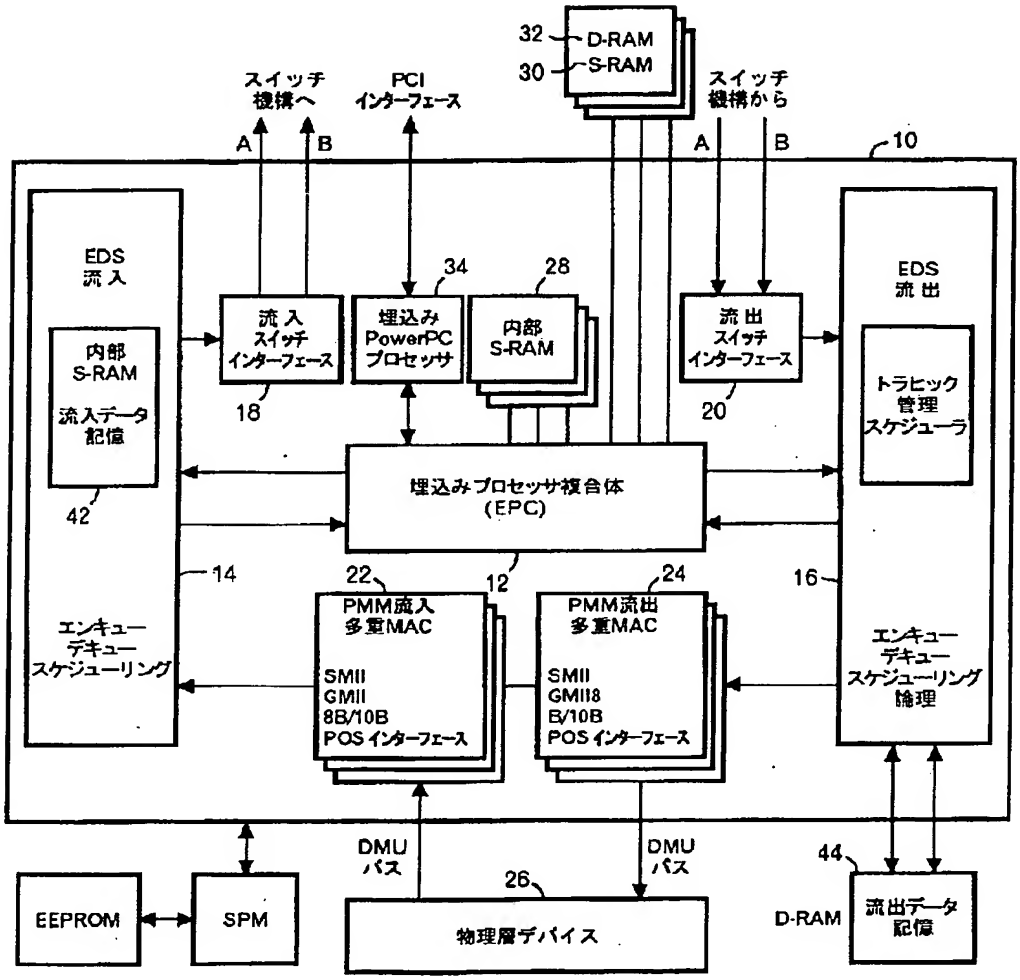
35

１６　エンキュー・デキュー・スケジューリング論理回路
１８　流入スイッチ・インタフェース
２０　流出スイッチ・インタフェース
２２　物理ＭＡＣマルチプレクサ
２４　物理ＭＡＣマルチプレクサ
２６　物理層デバイス

36

２８　内部ＳＲＡＭ
３０　外部ＺＢＴ　ＳＲＡＭ
３２　外部ＤＤＲ　ＤＲＡＭ
３４　制御ポイント・プロセッサ
４２　流入データ記憶
４４　流出データ記憶

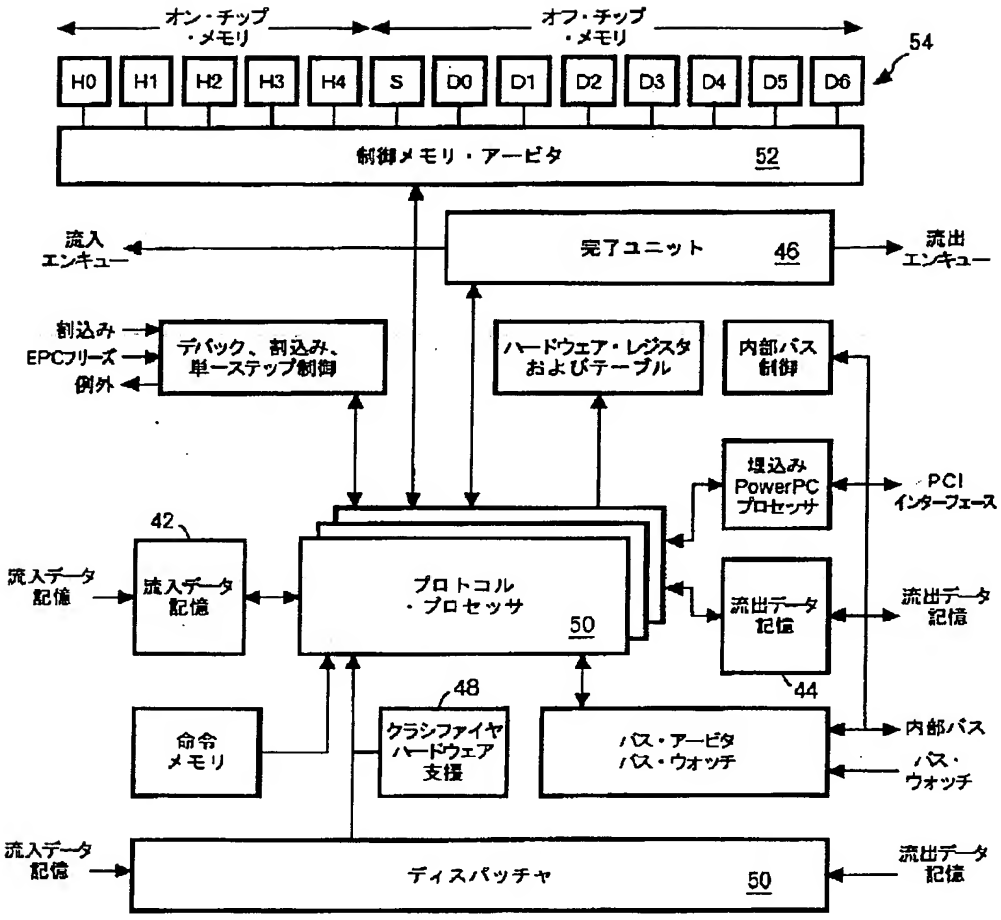
【図１】



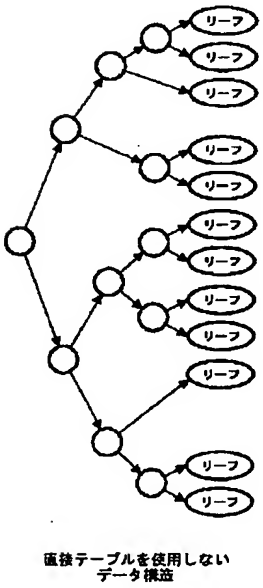
【図８】

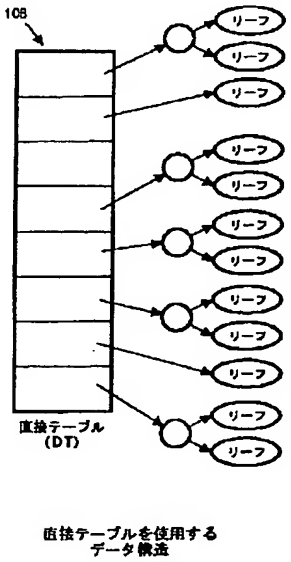
入力キー	IPSA (32 bits)	IPDA (32 bits)	ソース・ポート (16 bits)	宛先ポート (16 bits)	プロトコル (8 b)
リーフパターン 0	値	値	Min	Min	値
リーフパターン 1	マスク	マスク	Max	Max	マスク

【図２】

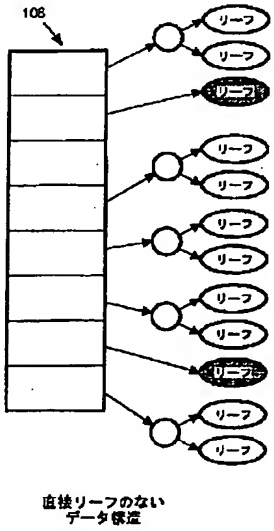

オン・チップ・メモリ
オフ・チップ・メモリ
54
H0
H1
H2
H3
H4
S
D0
D1
D2
D3
D4
D5
D6
制御メモリ・アービタ
52
流入エンキュー
完了ユニット
46
流出エンキュー
割込み
EPCフリーズ
例外
デバック、割込み、単一ステップ制御
ハードウェア・レジスタおよびテーブル
内部バス制御
埋込みPowerPCプロセッサ
PCIインターフェース
42
流入データ記憶
流入データ記憶
プロトコル・プロセッサ
50
流出データ記憶
流出データ記憶
44
48
命令メモリ
クラシファイヤハードウェア支援
バス・アービタバス・ウォッチ
内部バス
バス・ウォッチ
流入データ記憶
ディスパッチャ
50
流出データ記憶

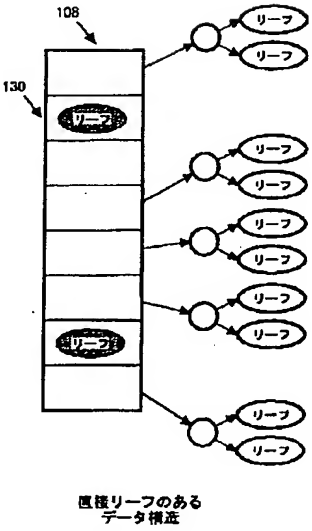
【図６】


リーフ
直接テーブルを使用しないデータ構造

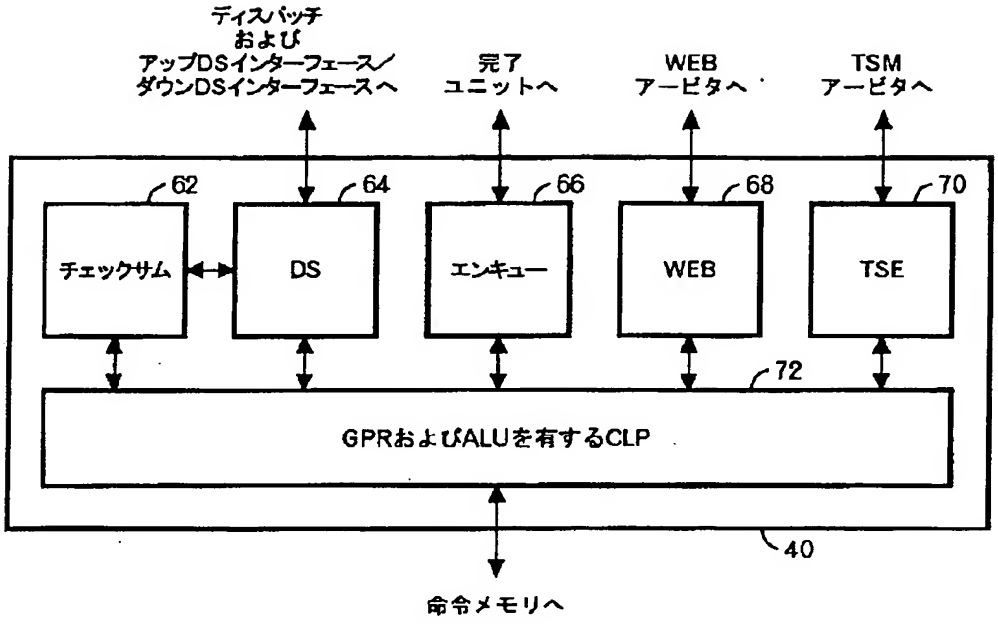

108
リーフ
直接テーブル(DT)
直接テーブルを使用するデータ構造

【図７】

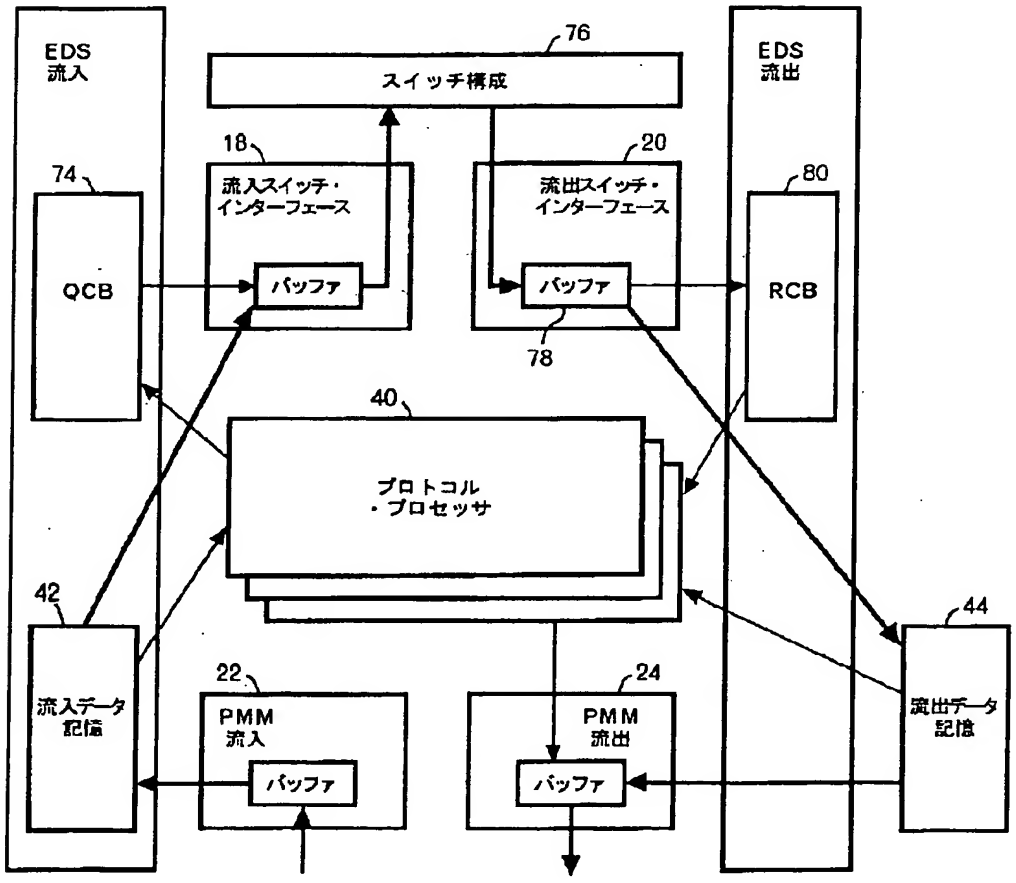

106
リーフ
直接リーフのないデータ構造


108
130
リーフ
直接リーフのあるデータ構造

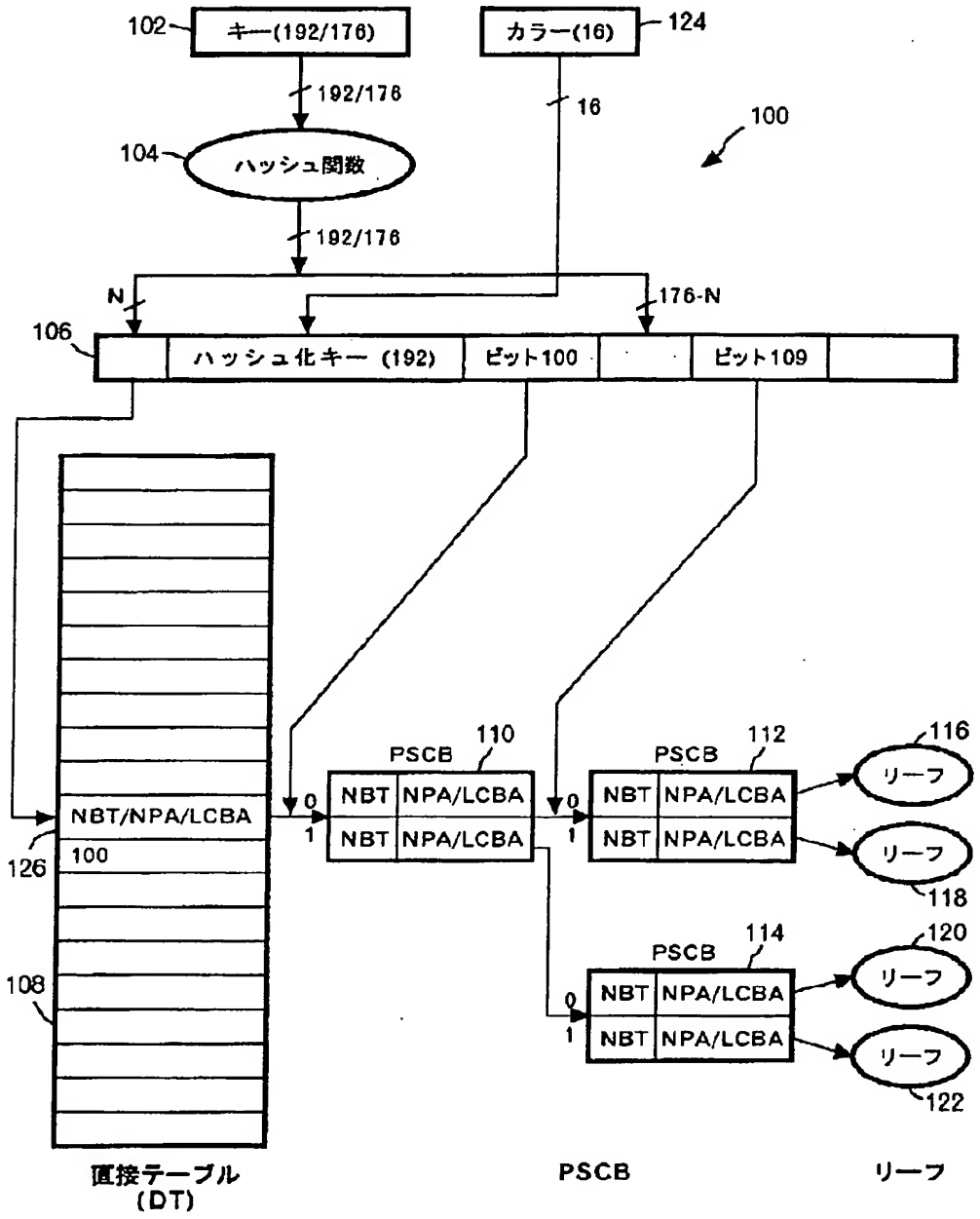
【図３】


ディスパッチ
および
アップDSインターフェース／
ダウンDSインターフェースへ
完了
ユニットへ
WEB
アービタへ
TSM
アービタへ
62
64
66
68
70
チェックサム
DS
エンキュー
WEB
TSE
72
GPRおよびALUを有するCLP
40
命令メモリへ

【図４】


EDS
流入
76
スイッチ構成
EDS
流出
18
流入スイッチ・
インターフェース
20
流出スイッチ・
インターフェース
74
QCB
バッファ
バッファ
80
RCB
78
40
プロトコル
・プロセッサ
42
44
流入データ
記憶
22
PMM
流入
バッファ
24
PMM
流出
バッファ
流出データ
記憶

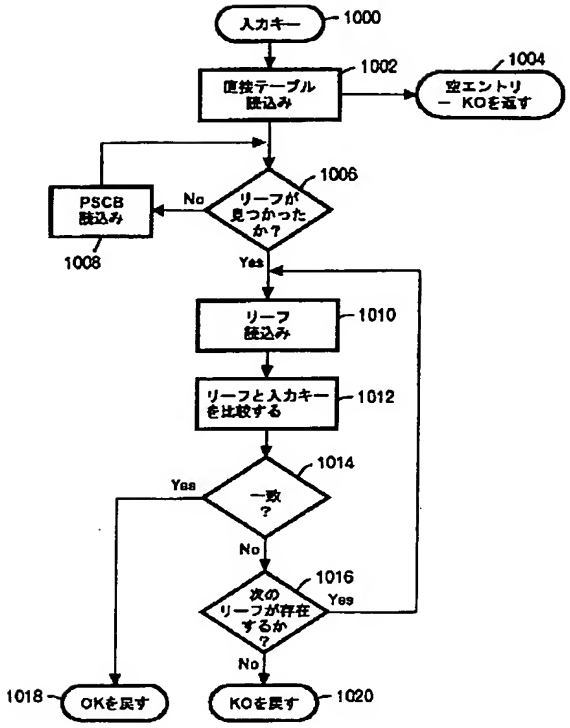
【図５】



【図９】

範囲１		範囲２			
オフセット フィールド0	最小最大 長さ フィールド0	オフセット フィールド1	最小最大 長さ フィールド1	１範囲／２範囲	継続
8 bits	8 b	8 bits	8 b	1 bit	1 bit
				０：範囲１が妥当 １：範囲１、２が妥当	０：この比較で停止 １：次の比較で継続

【図１０】



【図１１】

LUDefTableツリー定義

フィールド	サイズ	ビット
CacheEntry	1	0
Tree_Type	2	2..1
hash-type	4	6..3
color_en	1	7
P1P2_max_size	5	12..8
NPARope_en	1	13
NPASMT_en	1	14
CompIndex_en	1	15
PSCB_fq_index	6	21..16
PSCB_Height	1	22
Mask_Vector_En	1	23
CompIndex	8	31..24
DT_base_addr	26	57..32
DT_size	4	61..58
DT_interleaf	2	63..62
Leaf_fq_index	6	69..64
Leaf_Width	2	71..70
Leaf_Height	3	74..72
DirectLeafEn	1	75

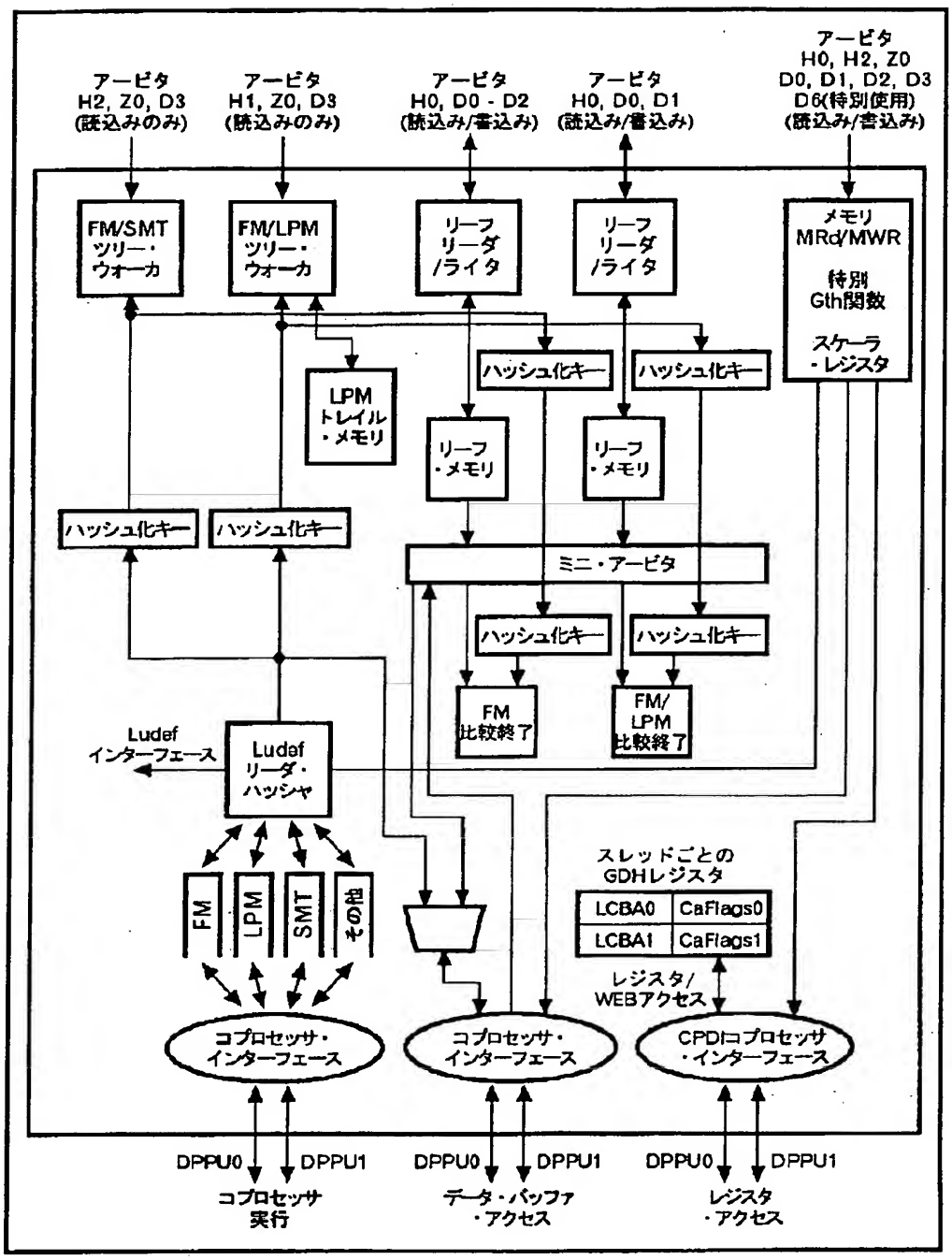
【図１２】

フィールド	サイズ	PSCBが配置されるTSM内のアドレス
NPA0	26	次のPSCBアドレス： PSCBの0-部分のための、ツリー内の次のPSCBに対するポインタ
NBT0	8	PSCBの0-部分のための、テストすべき次のビット
NCBA0	26	リーフ制御ブロック・アドレス： PSCBの0-部分のためのリーフに対するポインタ
NPA1	26	次のPSCBアドレス： PSCBの1-部分のための、ツリー内の次のPSCBに対するポインタ
NBT1	8	PSCBの1-部分のための、テストすべき次のビット
LCBA1	26	リーフ制御ブロック・アドレス： PSCBの1-部分のためのリーフに対するポインタ
Index	8	このPSCBのインデックス (前のPSCBに物理的に記憶されている)
PatBit	1	PSCBレジスタ内のインデックス・フィールドの 値に基づくHashedkey[インデックス]の値

【図１３】

フィールド名	長 さ	内 容
NLASMT	4 バイト	SMTのリーフを結びつけるためのリーフ連鎖ポインタ 結びつけられたリーフの形を含む
Comp_table_index	1 バイト	Pattern1、Pattern2とHashedKeyの間の比較を定義する CompDefTable内のインデックスを定義する
Pattern1 and Pattern2	4-36 バイト	このフィールドは、Pattern1およびPattern2を ビットごとに交互に含む すなわちこのフィールドのビット0はPattern1の ビット0を含み、ビット1はPattern2のビット0を 含む 交互に偶数ビットはPattern1を表し、奇数ビットは Pattern2を表す
UserData	可 変	このフィールドの内容は完全なピココード制御下にある； UserDataフィールドは１つまたは複数のカウンタを 含むことができる

【図１４】



フロントページの続き

(72)発明者　ジャン・ルイ・カルヴィニャク
アメリカ合衆国27511　ノースカロライナ
州ケアリー　スプリング・ホロー・レーン
112

(72)発明者　マーコ・シー・ヘデス
アメリカ合衆国27612　ノースカロライナ
州ローリー　グランド・マナー・コート
4109　ナンバー308

(72)発明者　クラーク・デブス・ジャフリーズ
アメリカ合衆国27713　ノースカロライナ州ダラム　ベインブリッジ・ドライブ　2806エイチ
(72)発明者　ピュシ・チュニラル・パテル
アメリカ合衆国27513　ノースカロライナ州ケアリー　スモークモント・ドライブ　105
(72)発明者　マーク・アンソニー・リナルディ
アメリカ合衆国27713　ノースカロライナ州ダラム　クィーンズベリー・サークル　1201

Ｆターム(参考)　5B075 KK02 ND02 ND34 ND35 NK13 NK24 NK43 NK45 PP22 QM01 QS13 UU16
5K030 KA01 KA02